The Journal of Systems and Software xxx (2011) xxx–xxx

Contents lists available at SciVerse ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

A documentation framework for architecture decisions

U. van Heesch[a,b,*], P. Avgeriou[a], R. Hilliard[c]

[a] University of Groningen, Groningen, The Netherlands
[b] Fontys University of Applied Sciences, Venlo, The Netherlands
[c] Consulting Software Systems Architect, Massachusetts, USA

ARTICLE INFO

Article history:
Received 1 May 2011
Received in revised form 15 October 2011
Accepted 17 October 2011
Available online xxx

Keywords:
Software architecture
Architecture decisions
Architecture knowledge management
Architectural viewpoints
Case study
Architecture framework

ABSTRACT

In this paper, we introduce a documentation framework for architecture decisions. This framework consists of four viewpoint definitions using the conventions of ISO/IEC/IEEE 42010, the new international standard for the description of system and software architectures. The four viewpoints, a Decision Detail viewpoint, a Decision Relationship viewpoint, a Decision Chronology viewpoint, and a Decision Stakeholder Involvement viewpoint satisfy several stakeholder concerns related to architecture decision management.

With the exception of the Decision Stakeholder Involvement viewpoint, the framework was evaluated in an industrial case study. The results are promising, as they show that decision views can be created with reasonable effort while satisfying many of the stakeholder concerns in decision documentation.

1. Introduction

With the growing complexity and size of software-intensive systems, software architecture has become increasingly important. While architecture is traditionally understood as the design of the system itself, manifested mainly in design elements and their form, Perry and Wolf recognized the importance of (design-)rationale as an integral part of the software architecture. They defined software architecture as follows:

Software Architecture = {*Elements*, *Forms*, *Rationale*} (Perry and Wolf, 1992)

Kruchten adopted this definition of software architecture as a starting point for the 4+1 View Model framework (Kruchten, 1995). In this framework, each of the five views addresses various system concerns and determines the organization of a set of architectural elements, the forms and patterns used, and the rationale behind those architectural choices. This concept of documenting software architecture as a set of views that correspond to viewpoint (VP) definitions and address system concerns was adopted and generalized in IEEE Std 1471:2000 (IEEE, 2000), and further elaborated by the architecture community (e.g., Clements et al., 2002; Rozanski and Woods, 2005). However, as in Kruchten's 4+1, the importance of documenting decisions and their rationale along with the selected architectural concepts was only mentioned, but little guidance was offered on how to document decisions.

Bosch emphasized the importance of documenting architecture as a set of architecture decisions (ADs) (Bosch, 2004). In contrast to the aforementioned approaches, design decisions as an explicit part of the software architecture description provide insight into the reasoning process and record the rationale behind design decisions. The concept of architecture decisions has been incorporated into ISO/IEC/IEEE 42010, (ISO, 2011), which is the international revision of IEEE Std 1471:2000 (IEEE, 2000).

Today, the perspective of looking at software architecture in terms of a set of architecture decisions is widely recognized. Authors have proposed templates for the information content that is important to capture about decisions (e.g., Jansen and Bosch, 2005; Tyree and Akerman, 2005), and various models and tools to capture and manage architecture decisions have been proposed (Tang et al., 2010). Several approaches incorporate the documentation of architecture decisions in architecture practice and subsequently capture and organize architecture decisions to address various concerns, such as traceability and architectural conformance (Babar et al., 2009).

There are currently three main approaches to documenting architecture decisions: *decision templates*, *decision models*, and *annotations*. We argue that all three approaches satisfy some decision-related concerns, but none of them succeeds in satisfying all concerns. Shortcomings of architecture decision documentation approaches are not surprising: as with traditional architecture

* Corresponding author at: University of Groningen, Groningen, The Netherlands.
E-mail addresses: uwe@vanheesch.net, u.vanheesch@fontys.nl (U. van Heesch), paris@cs.rug.nl (P. Avgeriou), r.hilliard@computer.org (R. Hilliard).

0164-1212/$ – see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2011.10.017

views, there is not a single way of documenting architecture decisions that frames all concerns of all stakeholders in an adequate and useful manner. We propose that multiple dedicated viewpoints should be defined that focus on framing specific decision-related concerns.

In this paper, we propose a *documentation framework* consisting of four viewpoints for architecture decisions: a Decision Detail viewpoint, a Decision Relationship viewpoint, a Decision Chronology viewpoint and a Decision Stakeholder Involvement viewpoint. Each viewpoint is dedicated to framing specific decision-related concerns. At the same time, each viewpoint is integrated with the other viewpoints through a common metamodel to offer a more complete picture of decisions and their rationale. The framework proposed here is useful "out of the box", but it can also serve as a basis for customization or extension by adding new decision-related viewpoints. One extension of the framework currently being investigated adds a viewpoint dedicated to decision-making forces (see Section 6). This viewpoint also builds upon the current framework metamodel. Apart from the Decision Stakeholder Involvement viewpoint, all viewpoints were validated in an industrial case study with very promising results.

The rest of this paper is organized as follows. Section 2 presents decision-related concerns. In Section 3, we briefly outline the proposed viewpoints including an example view for each of the viewpoints.[1] In Section 4, we report on an industrial case study, which was conducted to validate the viewpoints. Section 5 summarizes related work. Section 6 presents our conclusions and ideas for future work.

2. Concerns related to architecture decisions

Architecture decisions should be documented to complement architectural design with rationale. Yet, how to capture decisions is still subject to discussion. This is mainly because there is no consensus on which stakeholder concerns must be addressed by a decision documentation approach. A *concern*, as used here, is any interest in a system on the part of its stakeholders. Each concern poses a question or issue that the architecture description, in this case the architecture decision documentation, should be able to answer.

In recent years, many use cases for architectural knowledge management have been published in the literature. We argue that decisions are one type of architectural knowledge. Therefore, we have analyzed three recent publications containing architectural knowledge management (AKM) use cases (Liang et al., 2009; Kruchten et al., 2006; Jansen et al., 2007) to identify and derive concerns for architecture decision documentation. Table 1 shows the resulting concerns.[2] The concerns were functionally grouped and, where possible, ordered according to the authors' estimation of their importance. The actual importance of the concerns, however, often depends on the specific needs of the concrete stakeholder.

The analysis procedure, as well as a complete table with the analyzed use cases, the derived concerns, and the activities performed to derive the respective concerns can be found in Appendix A.

Next, the authors assigned the concerns to typical stakeholders. Table 2 shows the results. Most concerns were assigned to architects and reviewers, because these stakeholders are frequently using architecture documentation in their daily work. Note that the assignment of the concerns took place based on typical tasks that the stakeholders perform in software projects. It could be argued that requirements engineers or managers for instance could also be interested in dependencies between decisions or the impact of a change in the architecture. However, we decided to limit ourselves to the most characteristic concerns for the respective stakeholders.

The concerns were taken as a basis for the development of the decision viewpoints, which will be introduced in the following section. Each of the viewpoint definitions is driven by the typical stakeholders and concerns it frames. With the exception of the concerns that were exclusively assigned to the Stakeholder Involvement viewpoint (C16, C17, and C19), all concerns for the viewpoints were validated as part of the case study presented in Section 4.

Table 1
Concerns for architecture decision documentation.

Code	Concern
C1	What decisions have been made?
C2	What is the current set of relevant decisions?
C3	What is the rationale for decision D?
C4	What concerns C_i does decision D address?
C5	What forces impacted/influenced each decision?
C6	What decisions affect concern C?
C7	What decisions have conflicting impacts on concern C?
C8	What decisions are required by decision D (including unmade decisions)?
C9	What decisions conflict with decision D?
C10	What decisions are dependent on decision D?
C11	What decisions are related to decision D?
C12	What decisions influence decision D (or architecture element E)?
C13	What decisions are impacted by a change?
C14	What decisions would be impacted when integrating a set of decisions S?
C15	How to apply a set of decisions from a different project in the target architecture?
C16	Which stakeholders are affected by decision D?
C17	What decisions affect stakeholder S?
C18	Which stakeholders were involved in decision D?
C19	What decisions are influenced by stakeholder S?
C20	What is the ordering of decisions made?
C21	What decisions have changed since time T (or milestone M)?
C22	What decisions became obsolete after change CH?
C23	What decisions D or decision sub-graphs SG can be reused in other projects?

3. A framework for architecture decision documentation

An *architecture framework* is a set of practices for architecture description used within a domain or community of stakeholders (ISO, 2011). A framework typically consists of a set of viewpoints for addressing recurring or typical concerns within that community. Fig. 1 shows the metamodel for architecture frameworks from ISO/IEC/IEEE 42010, (ISO, 2011).

In this paper, we present a documentation framework for architecture decisions which uses the conventions of ISO/IEC/IEEE 42010. It comprises all elements defined in Fig. 1. The four viewpoints of the framework were successively developed to frame the concerns described in the previous section. Each of the viewpoints is dedicated to concerns that are not or not

Table 2
Architecture decision concerns related to typical stakeholders.

Stakeholders	Concerns
Architects	C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11, C12, C13, C14, C15, C16, C17, C18, C19, C20, C21, C22, C23
Reviewers	C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11, C12, C16, C18, C20, C21
Managers	C17, C18, C19
Customers	C3, C6, C7
Requirements engineers	C6, C7
New project members	C3, C20
Domain experts	C23

[1] The whole documentation framework in terms of a unified metamodel, the complete viewpoint definitions, and the correspondences between those viewpoints are specified in Appendix C.

[2] A decision sub-graph, as used in concern C23, is a subset of a bigger set of interrelated decisions.

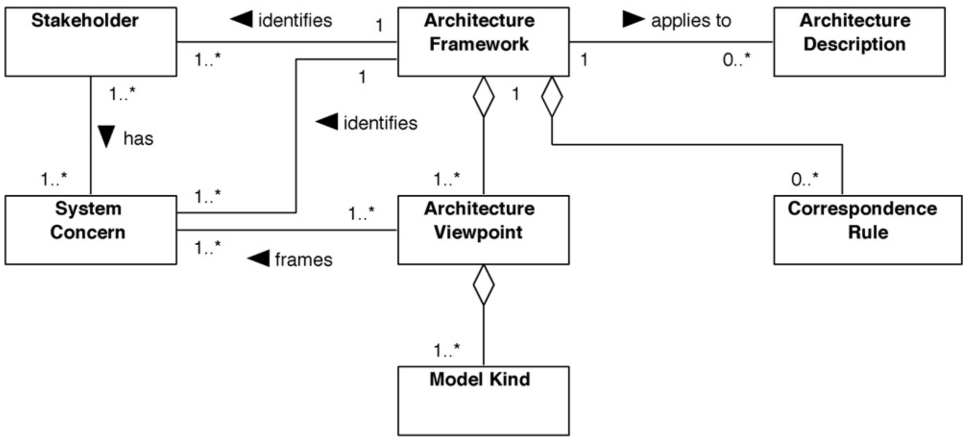


Fig. 1. Architecture framework (reproduced from ISO/IEC/IEEE 42010, ISO, 2011).

sufficiently framed by the previously created viewpoint. Starting from the Decision Detail viewpoint, which mainly addresses concerns related to the rationale behind decisions (C3–C6); we defined the Decision Relationship viewpoint, which focusses on concerns pertaining to relationships between decisions (C8–C15). The Decision Stakeholder Involvement viewpoint allows one to explicate the relationships between stakeholders and decisions (C16–C19). Finally, the Decision Chronological viewpoint was developed to satisfy the remaining temporal concerns in decisions (C20–C22). Apart from the key concerns mentioned here, each viewpoint addresses additional concerns that will be described in the following subsections.

In the remainder of this section, we outline the four viewpoints and show example views. A thorough definition of the framework can be found in Appendix C.

3.1. Decision Detail viewpoint

Although textual decision descriptions have disadvantages as mentioned in the introduction, they are certainly useful for grasping large parts of the rationale behind decisions. We propose to complement the previously described viewpoints, which only include partial information about the decisions, with a Decision Detail viewpoint. Every viewpoint frames specific stakeholder concerns, omitting information that is irrelevant for the respective stakeholders or those specific concerns. While the other viewpoints provide an overview over the decisions made and focus on the relationships between decisions, the Decision Detail viewpoint gives detailed information about single decisions.

Currently, there is no commonly accepted template in the literature to describe architecture decisions, although many proposals exist. Shahin et al. analyzed nine architecture decision models with respect to similar description elements (Shahin et al., 2009). They distinguish between major (e.g., decision outcome, related requirements, design options, arguments) and minor description elements (e.g., issue, decision group, state, related decisions, related artifacts, consequences and stakeholders). We decided to create our own template that contains all major elements plus selected minor elements that turned out to be useful in three pilot studies we conducted to test the decision viewpoints. The resulting set of description elements is:

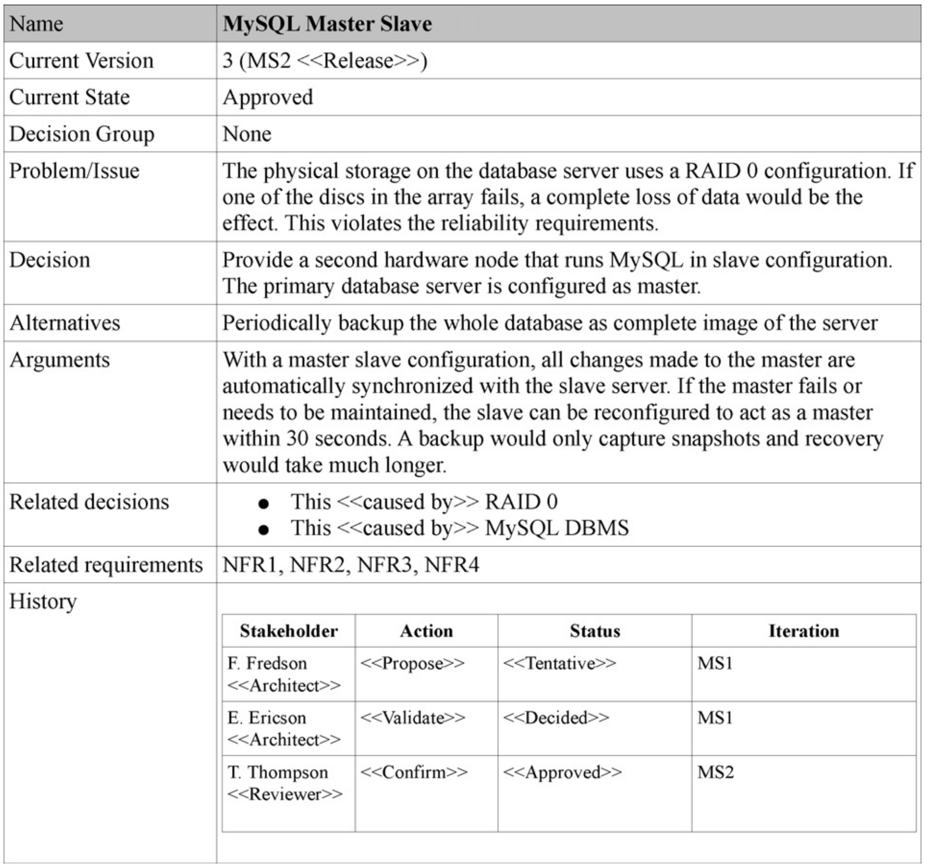
- **Name**: A short name of the decision that serves as a key in the other views.
- **Current state**: The current state of the decision. Please refer to Fig. C.3 for a list of all possible decision states.
- **Decision groups**: A decision can be associated to one or more groups, which share specific characteristics. Decisions could for instance be grouped by subsystem, architecture team who made the decision, or quality attribute requirements. The concept of a decision group is equal to the group concept in Tyree and Akerman's decision template (Tyree and Akerman, 2005), and the decision categories in Kruchten's ontology (Kruchten, 2004).
- **Problem/issue**: The circumstances under which the architect felt the need to make a decision among one or more alternatives. In other words, the issue addressed by the decision.
- **Decision**: The outcome of the decision. In other templates this element is called *solution*.
- **Alternatives**: The alternative solutions considered when making the decision.
- **Related decisions**: All decisions that have a relationship to the decision. The available relationship types are defined in Appendix C.2.
- **Related system concerns**: The term *system concern* is taken from ISO/IEC/IEEE 42010, (ISO, 2011), describing any interest in a system on behalf of its stakeholders. System concerns include, among others: functional and non-functional requirements, constraints, business goals, assumptions, risks, and design rules. The Decision Detail viewpoint is currently the only viewpoint that allows one to trace requirements and architecture decisions. An additional viewpoint that specializes on traceability between requirements and decisions will be discussed in Section 6.
- **History**: The history of the described decision. The history contains all state changes, i.e., when the decision was proposed, decided, approved and so on.

Typical stakeholders for this viewpoint are reviewers, architects, customers, managers, new project members and requirements engineers. Table 3 shows the concerns framed by this viewpoint in relation to the respective stakeholders.

Writing elaborate decision descriptions is a resource-intensive task. However, the flexibility of this viewpoint allows companies to document just as much as needed for their individual purposes.

Table 3
Typical stakeholders of the Decision Detail viewpoint.

Stakeholders	Concerns
Reviewers	C1, C2, C3, C4, C5, C6, C11, C18
Architects	C1, C2, C3, C4, C5, C6, C11, C18
Customers	C3, C6
Managers	C18
New project members	C3
Requirements engineers	C6



Name	**MySQL Master Slave**
Current Version	3 (MS2 <<Release>>)
Current State	Approved
Decision Group	None
Problem/Issue	The physical storage on the database server uses a RAID 0 configuration. If one of the discs in the array fails, a complete loss of data would be the effect. This violates the reliability requirements.
Decision	Provide a second hardware node that runs MySQL in slave configuration. The primary database server is configured as master.
Alternatives	Periodically backup the whole database as complete image of the server
Arguments	With a master slave configuration, all changes made to the master are automatically synchronized with the slave server. If the master fails or needs to be maintained, the slave can be reconfigured to act as a master within 30 seconds. A backup would only capture snapshots and recovery would take much longer.
Related decisions	● This <<caused by>> RAID 0 ● This <<caused by>> MySQL DBMS
Related requirements	NFR1, NFR2, NFR3, NFR4
History	

Stakeholder	Action	Status	Iteration
F. Fredson <<Architect>>	<<Propose>>	<<Tentative>>	MS1
E. Ericson <<Architect>>	<<Validate>>	<<Decided>>	MS1
T. Thompson <<Reviewer>>	<<Confirm>>	<<Approved>>	MS2

Fig. 2. Example detail model of an architecture decision.

Some organizations might even decide to skip this viewpoint completely and only use some of the other proposed viewpoints to document key aspects of their decisions. Others might decide to use a subset of the proposed elements of our template. The project team in our case study felt that there is no need to document every decision in the same level of detail. They described the major and most important decisions in detail, while putting less effort in describing minor decisions.

3.2. *Decision Relationship viewpoint*

The Decision Relationship viewpoint makes relationships between architectural design decisions explicit. It shows architecture decisions, their relationships to other decisions, and their current states. It has no temporal component, i.e., it shows a snapshot of the system in a particular moment in time. Typical stakeholders for this viewpoint are architects, reviewers and domain experts. Table 4 shows the concerns framed by the viewpoint as related to the mentioned stakeholders. They center mainly around impact, dependency and relationship analysis. In addition, relationship views are well-suited for getting an overview over all decisions made. Please refer to Table 1 for the descriptions of the concerns.

Table 4
Typical stakeholders and concerns for the Decision Relationship viewpoint.

Stakeholders	Concerns
Architects	C1, C2, C8, C9, C10, C11, C12, C13, C14, C15, C22, C23
Reviewers	C1, C2, C9, C10, C11, C12
Domain experts	C23

Fig. 3 shows an extract from a relationship view that was created in a preliminary study conducted to test the decision viewpoints. The preliminary studies are further described in Section 4.

3.3. *Decision Stakeholder Involvement viewpoint*

The Decision Stakeholder Involvement viewpoint shows the responsibilities of relevant stakeholders in the decision-making process. Views resulting from this viewpoint have no temporal component. They show decisions, actions and stakeholders involved in the decision-making process within one specific architecture iteration. This information is important with regard to personalization of architectural knowledge, i.e., documenting not the knowledge per se but who knows what. For many reasons, in some projects, it is not feasible to fully document the rationale behind all architecture decisions. Other knowledge remains tacit; it is not documented at all. In these situations, the rationale remains in the heads of the people who were involved in the decision making process. Stakeholder involvement views make these involvements explicit. Furthermore, the viewpoint allows one to analyze the impact of personnel on the success or failure of a project. If, for instance, a large number of decisions made by one specific architect were rejected, then this could be an indicator for a problem. As a side effect, explicitly documenting responsibilities creates accountability, in that people assume responsibility for the decisions they are involved in. On the other hand, this might cause architects to neglect the usage of stakeholder views, because they fear accountability.

Typical stakeholders for this viewpoint are reviewers, architects and managers. Table 5 shows the concerns framed by this viewpoint related to the respective stakeholders. They center around

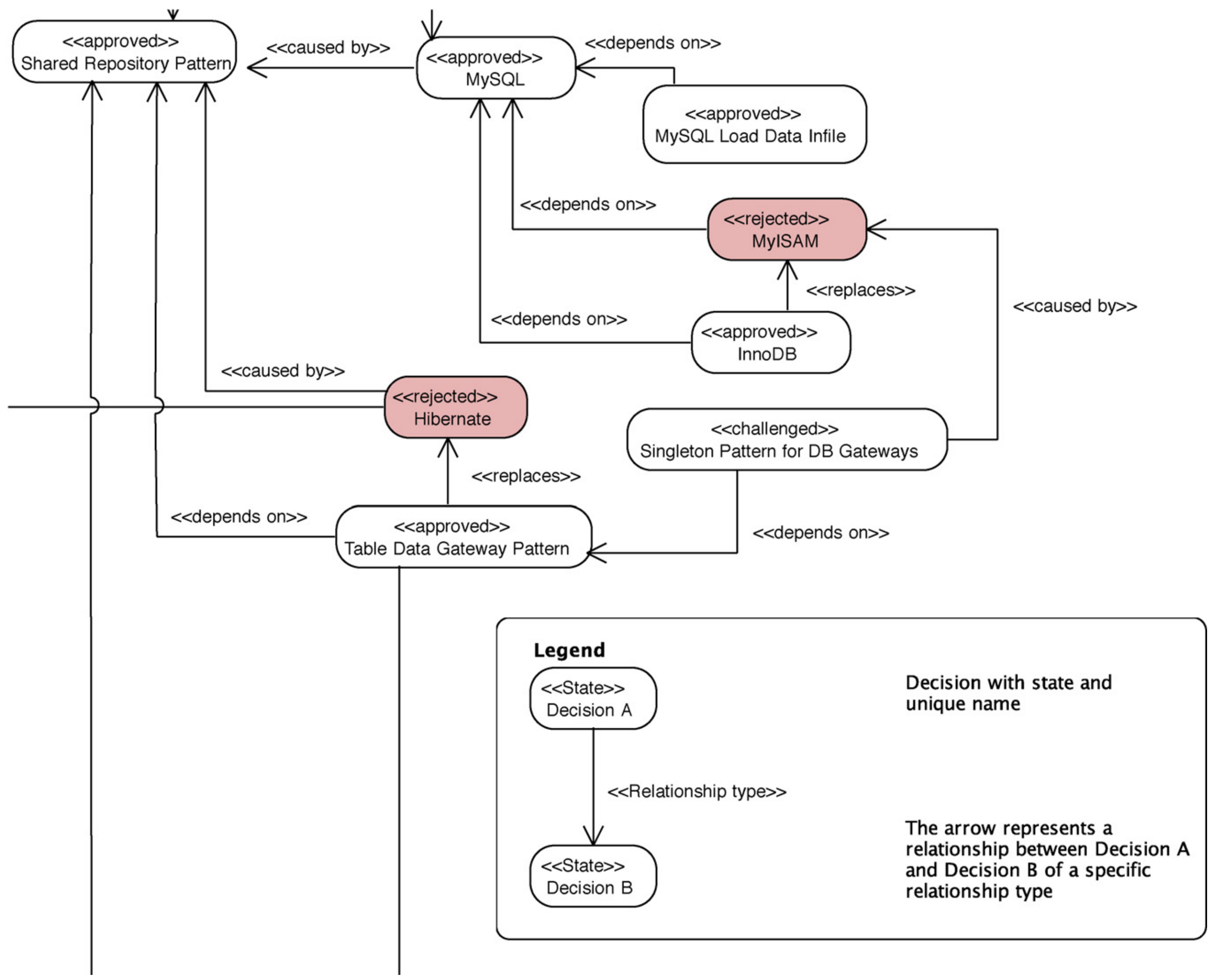

<<approved>> Shared Repository Pattern
<<caused by>>
<<approved>> MySQL
<<depends on>>
<<approved>> MySQL Load Data Infile
<<depends on>>
<<rejected>> MyISAM
<<replaces>>
<<caused by>>
<<depends on>>
<<approved>> InnoDB
<<caused by>>
<<rejected>> Hibernate
<<challenged>> Singleton Pattern for DB Gateways
<<replaces>>
<<depends on>>
<<approved>> Table Data Gateway Pattern
<<depends on>>
Legend
<<State>> Decision A
Decision with state and unique name
<<Relationship type>>
<<State>> Decision B
The arrow represents a relationship between Decision A and Decision B of a specific relationship type

Fig. 3. Extract of a relationship view.

stakeholder involvement in decisions. Please refer to Table 1 for the descriptions of the concerns.

Fig. 4 shows an extract from a stakeholder involvement view that was created in a preliminary study conducted to test the decision viewpoints. The names of the involved stakeholders were changed for privacy reasons. The preliminary studies are further described in Section 4.

3.4. *Decision Chronological viewpoint*

The Decision Chronological viewpoint shows the evolution of architecture decisions in chronological order. Besides decisions, it shows architecture iterations and its endpoints, which can be further specified by a type and a date. The chronological viewpoint is the only proposed decision viewpoint that has a temporal component. Typical stakeholders for this viewpoint are reviewers, architects and new project members who try to comprehend the architecting process during system evolution. Table 6 shows the concerns framed by this viewpoint related to the respective stakeholders. A chronological view shows all versions of every architecture decision of a system. A version of an architecture decision is defined as a decision with a state. For instance a decision that was *tentative*, then became *decided* and finally *approved* is represented with three instances in one chronological view.

Fig. 5 shows an extract from a chronological view created in the case study, which is presented in Section 4.

4. A case study

To validate the usage of the presented architecture decision viewpoints in a real software project, we conducted a single case, embedded case study (Gray, 2009). In a *single case* design, only a single case is observed; *embedded* refers to the fact that multiple units of analysis are observed in one case.

Table 5
Typical stakeholders of Decision Stakeholder Involvement viewpoint.

Stakeholders	Concerns
Reviewers	C16, C17, C18
Architects	C1, C16, C17, C18, C19
Managers	C18, C19

Table 6
Typical stakeholders of Decision Chronology viewpoint.

Stakeholders	Concerns
Reviewers	C1, C2, C20, C21
Architects	C1, C2, C20, C21, C22
New project members	C20

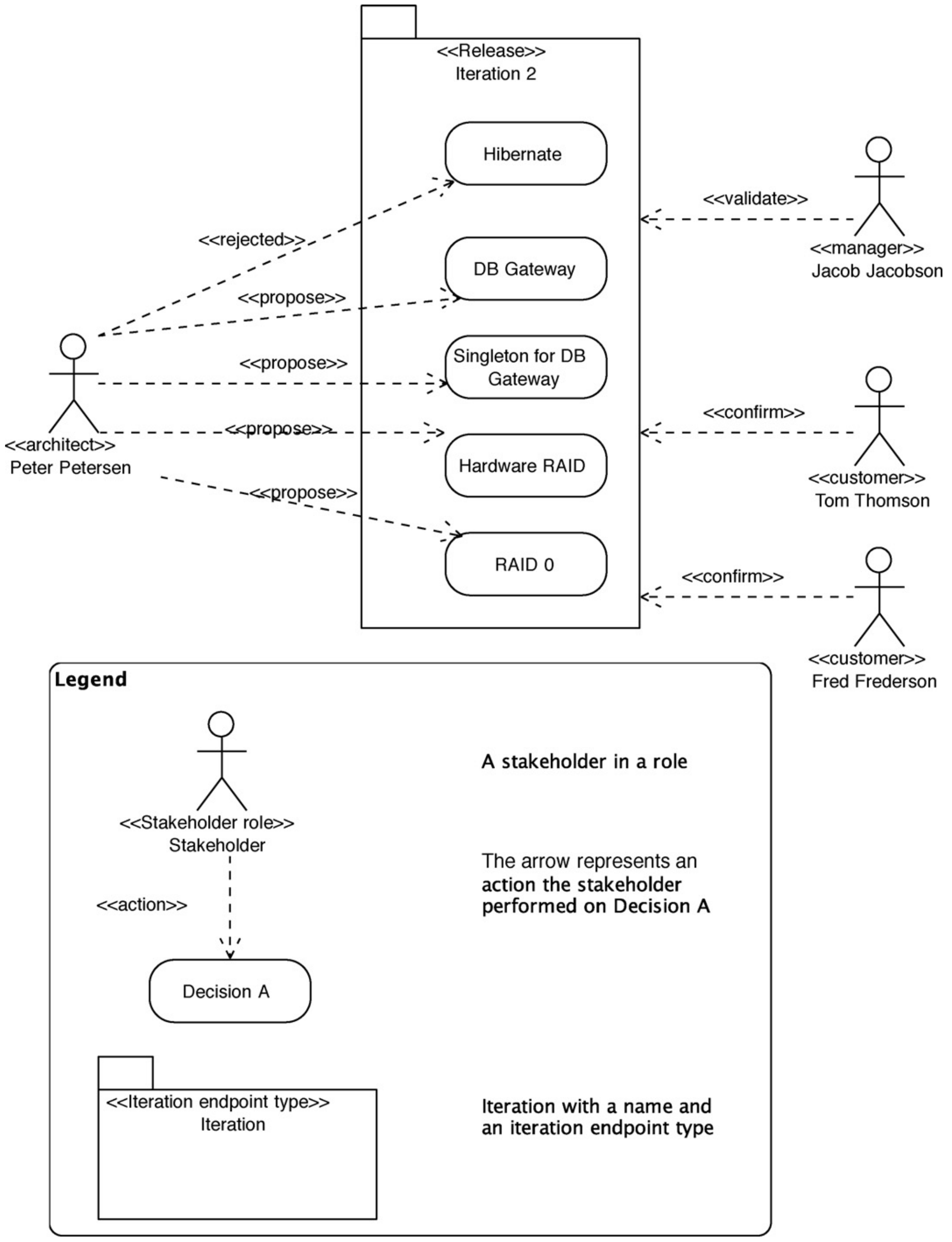


Fig. 4. Example stakeholder involvement view on architecture decisions.

This case study is a project executed at the Institute for Internet-Security (IFIS). The IFIS is a German organization in the Internet and network-security domain. We examined the decision viewpoints as applied to a software project called "Sandnet" (Rossow et al., 2011). Sandnet is a system that executes malware like viruses, worms and bots in a controlled environment to analyze their network behavior.

4.1. Study goal, research questions and variables

The goal of the case study is to explore whether the architecture decision viewpoints effectively support software architecture activities. To derive concrete research questions, we explained the decision viewpoints to the architects of the Sandnet project and let them decide for which architecture activities they could be used in their project. They identified the following activities: general architecture decision documentation, communication between stakeholders, technical architecture reviews, and reusing architecture decisions (i.e., architectural solutions) in other projects.

Then the architects of Sandnet expressed their concerns in decision documentation with respect to these activities. These concerns were mapped to the list of concerns shown in Table 1 and supplemented by concerns that the authors found important. Finally, the authors assigned to the activities all viewpoints that were designed to frame at least one of the concerns mentioned by the architects. Table 7 shows the results.

Based on the information shown in Table 7, the architects decided to create views according to the Decision Relationship

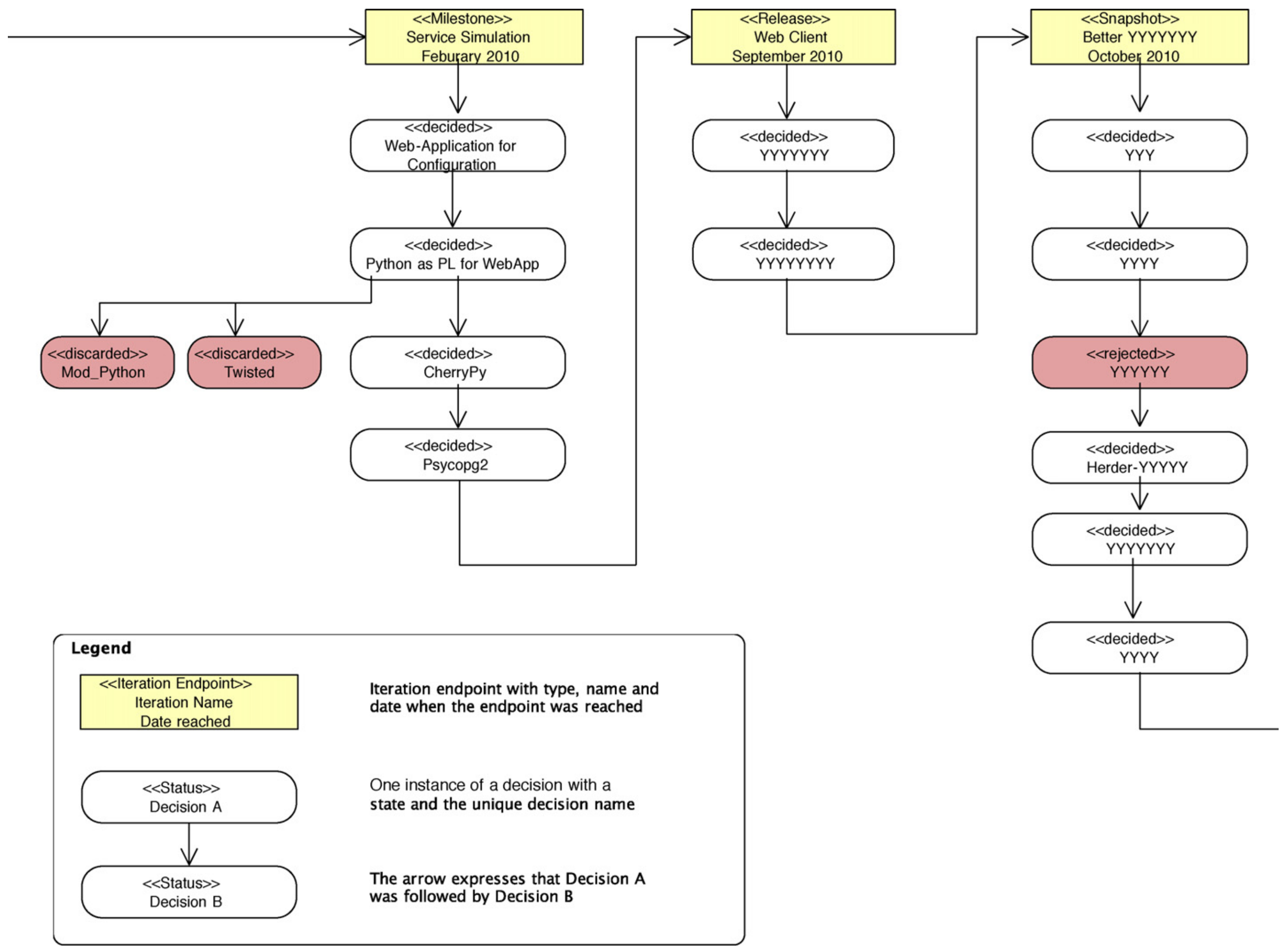


Fig. 5. Example from a chronological view of architecture decisions.

viewpoint, the Decision Chronological viewpoint, and the Decision Detail viewpoint. They did not see additional benefit in documenting a Stakeholder Involvement view, because they were convinced that the project was small enough to remember the involvement of all stakeholders. This is in-line with ISO/IEC/IEEE 42010, which propagates the choice of viewpoints to architects using the standard, according to the prioritization of the concerns. The concerns C16, C17, and C19 are not covered in the case study, because the are exclusively satisfied by the stakeholder involvement viewpoint. Concern C7 (*What decisions have conflicting impacts on concern C?*) is currently not covered by any of the viewpoints in the framework. We reflect on this issue in Section 6.

Next, we formulated concrete research questions matching the architecture activities selected by the architects and the related viewpoints.

The four resulting research questions are summarized in Table 8 and discussed in the remainder of this section.

4.1.1. *RQ1 – What is the effort of documenting architecture decisions using architecture decision viewpoints?*

Research question one (RQ1) is about the effort that architects have to make in order to document architecture decisions using decision viewpoints. Note that this question is more specific than the question "Do decision views effectively support stakeholders to document architecture decisions?", as could be derived from the main research goal applied to activity A1. We made RQ1 more specific, as the effort is essential to judge the effectiveness of the decision viewpoint approach, and it can be explicitly measured. RQ1 can be refined with respect to the different viewpoints under study, i.e.,

- What is the effort of creating a decision relationship view that conforms to the Decision Relationship viewpoint?
- What is the effort of creating a decision chronology view that conforms to the Decision Chronology viewpoint?

Table 7
Architecture activities, related concerns and viewpoints.

Architecture activity	Concerns	Viewpoints
A1 – AD documentation	All	All
A2 – Stakeholder communication	C1, C2, C3, C4, C5, C6, C8, C9, C10, C11, C12, C17, C19, C20, C21	All
A3 – Technical architecture reviews	C1, C2, C3, C5, C6, C7, C8, C9, C10, C11, C12, C16, C17, C18, C20, C21	All (C7 not covered)
A4 – Reusing ADs	C3, C5, C8, C17, C20, C23	Relationship viewpoint, Chronological viewpoint, Details viewpoint

Table 8
Architecture activities and related viewpoints.

Code	Research question	Viewpoints
RQ1	What is the effort of documenting architecture decisions using architecture decision viewpoints?	Relationship VP, Chronology VP, Details VP
RQ2	Do decision views effectively support stakeholders to understand the architecture?	Relationship VP, Chronology VP, Details VP
RQ3	Do decision views effectively support architecture reviews?	Relationship VP, Chronology VP, Details VP
RQ4	Do decision views support architects to distill reusable decision sub-graphs?	Relationship VP, Chronology VP, Details VP

Table 9
Dependent variables of RQ1.

Description	Scale type	Unit	Range
Time spent to create view	Ratio	Person-hours	Positive natural numbers including zero

- What is the effort of creating a decision detail view that conforms to the Decision Detail viewpoint?

One dependent variable is defined for RQ1: the effort of creating each of the views is measured in person-hours, a common unit to express effort in software development projects, here defined as the number of hours spent by one person. Table 9 summarizes the variables. The statistical scale used to measure values for the variable is a ratio scale, which means that the possible values for the variable are ordered, have a zero point, and have equal intervals (Wohlin et al., 2000). The scale type is needed to determine which statistical calculations apply for the variable. The range shows which values can actually be assigned to the variable. In this case, the time is measured in whole hours; thus, the variable can take positive natural numbers of hours including zero hours.

Table 10 shows potential variables that might have an influence on the effort needed to document the views. These independent variables relate to characteristics of the architects who used the viewpoints to create views, and to the characteristics of the software project that was documented. It is in the nature of case studies that these variables cannot be controlled (Gray, 2009), thus we describe them thoroughly so readers can use them to judge the external validity of the results. This and other potential threats to validity are discussed in Section 4.4.5.

4.1.2. RQ2 – Do decision views effectively support stakeholders to understand the architecture?

In research question two (RQ2), we want to find out if decision views, corresponding to the decision viewpoints, support stakeholders to understand the architecture. RQ2 is decomposed with respect to the different viewpoints under study:

- Do decision relationship views effectively support stakeholders to understand the architecture?
- Do decision chronology views effectively support stakeholders to understand the architecture?

Table 11
Dependent variables RQ2.

Description	Scale type	Unit	Range
Level of architecture understanding by the stakeholders	n.a.	Open	Open

- Do decision detail views effectively support stakeholders to understand the architecture?

The level of understanding of an architecture that stakeholders gain after studying the decision views is hard to measure and especially hard to quantify. As dependent variable, we estimate the level of understanding by qualitatively analyzing questions asked and comments expressed by stakeholders to the architects after having studied the decision views (see Table 11).

Table 12 shows independent variables that could influence the dependent variables. They relate to characteristics of the stakeholders who studied the views and the characteristics of the software project that was documented.

4.1.3. RQ3 – Do decision views effectively support architecture reviews?

Research question three (RQ3) aims at finding out if views, corresponding to the decision viewpoints, support activities performed during architecture reviews. According to IEEE (2008), a review is an evaluation of a software product by a team of qualified personnel. Accordingly, an architecture review is an evaluation of the software architecture by stakeholders who are either domain experts or architecture experts. In the case study, we had the opportunity to observe the usage of decision views in an architecture review. Details on the architecture review are given in Section 4.3.3.

Unfortunately, the IFIS organization had not been following an established or systematic review approach before the case study. The architects of the Sandnet project previously performed reviews in an ad hoc manner, without involving other stakeholders and without systematically documenting review outcomes. Thus, the effect of using decision views in the review cannot be compared to the previous practice.

RQ3 is decomposed with respect to the different viewpoints under study:

- Do decision relationship views effectively support architecture reviews?

Table 10
Independent variables of RQ1.

Description	Scale type	Unit	Range
Time the architects have worked in the IT industry	Ratio	Years	Positive natural numbers including zero
Time the architects have worked as software designers/architects	Ratio	Years	Positive natural numbers including zero
Number of architects who created the views	Ratio	Persons	Positive natural numbers including zero
Duration of the documented project	Ratio	Months	Positive natural numbers including zero
Project size	Ratio	Person-months	Positive natural numbers including zero
Number of made decisions	Ratio	Decisions	Positive natural numbers including zero
Average number of words used to document one decision in the detail view	Ratio	Words	Positive natural numbers including zero

Table 12
Independent variables RQ2.

Description	Scale type	Unit	Range
Time the stakeholder has worked in the IT industry	Ratio	Years	Positive natural numbers including zero
Time the stakeholder has worked as software designer/architect	Ratio	Years	Positive natural numbers including zero
Time the stakeholder has worked in the network security domain	Ratio	Years	Positive natural numbers including zero
How often has the stakeholder been involved in the analysis of architecture decisions	Ordinal	n.a.	Five point Likert-scale. One for very frequently, five for very rarely
Time the stakeholders took to study the views	Ratio	Min	Positive real numbers including zero
Duration of the documented project	Ratio	Months	Positive natural numbers including zero
Number of made decisions	Ratio	Decisions	Positive natural numbers including zero
Average number of words used to document one decision in the detail view	Ratio	Words	Positive real numbers including zero
Time spent to study a view	Ratio	Person-hours	Positive natural numbers including zero

Table 13
Dependent variables RQ3.

Description	Scale type	Unit	Range
Level of support for the review activities	n.a.	Open	Open
Number of risks uncovered during the review	Ratio	Risks	Positive natural numbers including zero

Table 15
Dependent variables RQ4.

Description	Scale type	Unit	Range
Number of identified reusable decisions	Ratio	Decisions	Positive natural numbers including zero
Level of support for identifying reusable decision sub-graphs	n.a.	Open	Open

- Do decision chronological views effectively support architecture reviews?
- Do decision detail views effectively support architecture reviews?

Two dependent variables are defined for RQ3. The level of support for the review activities is estimated by qualitatively analyzing the transcript from a focus group, conducted with the participants of an architecture review performed as part of the case study. We also measure the number of risks that came up during the review (see Table 13). An elaboration of the focus group can be found in Section 4.2.2.

Table 14 shows independent variables that could influence the suitability of decision views to support architecture reviews. They relate to characteristics of the people who took part in the review, the software project, and the review approach that is being followed.

4.1.4. *RQ4 – Do decision views support architects to distill reusable decision sub-graphs*

The last research question (RQ4) is about identifying a set of decision sub-graphs or logically grouped architecture decisions that can be reused as a whole in other software projects. An example for such a decision sub-graph is the choice of a database management system (DMBS), the choice of a hardware platform for the DBMS, the choice of an operating system for the hardware platform and a communication protocol for accessing the DBMS. The sub-graph contains all possible design options, the chronological order of the decisions and the rationale behind each of the decisions. RQ4 is decomposed with respect to the different viewpoints under study:

- Do decision relationship views support architects to distill reusable decision sub-graphs?
- Do decision chronological views support architects to distill reusable decision sub-graphs?
- Do decision detail views support architects to distill reusable decision sub-graphs?

To find out if decision views support this process, we independently asked the architects and other technical stakeholders to identify concrete reusable decisions. Subsequently, we counted the decisions and evaluated the level of support provided by the views in order to identify them (see Table 15).

Table 16 shows independent variables that might have an influence on the dependent variables. They relate to characteristics of the architects who distilled the reusable decisions, and the software project.

4.2. *Study design and execution*

The aim of case studies in general is the investigation of contemporary phenomena in their natural context (Robson, 2002; Yin, 2003). According to Robson's classification scheme for empirical research purposes (Robson, 2002), our case study is exploratory in nature. We aim at understanding how and which architecture

Table 14
Independent variables RQ3.

Description	Scale type	Unit	Range
Time the reviewers have worked in the IT industry	Ratio	Years	Positive natural numbers including zero
Time the reviewers have worked as software designer/architect	Ratio	Years	Positive natural numbers including zero
Time the reviewers have worked in the network security domain	Ratio	Years	Positive natural numbers including zero
How often have the reviewers been involved in the analysis of architecture decisions	Ordinal	n.a.	Five point Likert-scale. One for very frequently, five for very rarely
How often have the reviewers been involved in architecture reviews	Ordinal	n.a.	Five point Likert-scale. One for very frequently, five for very rarely
Activities performed in the architecture review	n.a.	Open	Open
Duration of the documented project	Ratio	Months	Positive natural numbers including zero
Number of decisions documented	Ratio	Decisions	Positive natural numbers including zero
Average number of words used to document one decision in the detail view	Ratio	Words	Positive real numbers including zero

Table 16
Independent variables RQ4.

Description	Scale type	Unit	Range
Time the stakeholders have worked in the IT industry	Ratio	Years	Positive natural numbers including zero
Time the stakeholders have worked as software designers/architects	Ratio	Years	Positive natural numbers including zero
Number of made decisions	Ratio	Decisions	Positive natural numbers including zero

activities can be supported by decision viewpoints. We chose the case study method because we aim at validating the proposed documentation framework in industrial practice, where the researchers' control on the observed events is typically low. This is in line with Gray, who suggests that case studies are appropriate where "how" and "why" questions about a set of events must be answered, over which the researcher has no control (Gray, 2009). Additionally, the effect of using decision viewpoints in a project might be multifaceted, which also makes the case study method more suitable than other empirical research methodologies which require control over independent variables (e.g., a controlled experiment) (Wohlin et al., 2003).

To elicit hidden variables and as a preparation for the study design, we conducted three small pilot studies with the decision viewpoints. In all three projects, our viewpoints were used to document the design decisions made. One of the authors was involved in all of the projects. The following projects were used as pilot studies:

- Open Pattern Repository (OPR): a freely usable, open source online repository for patterns and technologies. All project artifacts including source code, architecture decisions and design documents can be found in the project's Google code repository (Code-Google, 2011).
- Open Decision Repository (ODR): the ODR is an open source web documentation tool for architecture and design decisions. Like the OPR, all project artifacts can be found in the project's Google code repository (Opendecisionrepository, 2011). We elaborate on the Open Decision Repository in Section 6.
- Measurement collector for network traffic analysis: this so-called "raw data transfer system" is part of an Internet early warning suite. The part of the system we looked at is responsible for collecting data measured by probes that are located in different autonomous systems that comprise the German connection to the Internet. Details about the vendor as well as the software itself cannot be provided, as the organization asked us to treat this data confidentially.

In the following subsections, the observed case and the used data collection methods are described.

4.2.1. *Case description*

The IFIS, in which the case study was conducted, currently has 49 employees working in nine main projects (as of March 2011). The project domains include cloud computing, botnet analysis, identity management and Internet early warning. Customers of the organization are, among others, large telecommunication providers and the German Federal Office for Information-Security.

In the project under study (Sandnet), malware collected on the Internet is executed in a prepared network for further analysis. The software provides a controlled execution environment for the extensive analysis and safe execution of malware samples. One of the major challenges in the project is to provide a realistic environment for malware execution on the one hand, while preventing the malware from doing harm in external networks. For instance, the Sandnet forwards denial-of-service or spam attacks to a dedicated honeypot server to protect the original destination of the attack, while analyzing the complete network traffic. The project started in September 2009 and is ongoing. In total, four developers are involved in the project; two of them are responsible for the software architecture. Important stakeholders of the system, apart from the developers and architects, are network administrators who operate the Sandnet in their networks and malware authors who have a negative stake in the system, because they do not want their malware to be analyzed. The project team follows no defined software development process. They work in small iterations of a few weeks and document the system in a company-wide wiki.

4.2.2. *Data collection*

Data gathered in case studies is mainly qualitative. Because qualitative data is typically less precise than quantitative data, it is important to use triangulation to increase the precision of the study (Runeson and Hoest, 2009). Triangulation provides a broader representation of the research object under study. We use two different types of triangulation: methodological and data source triangulation (Stake, 1995). Methodological triangulation takes different types of data collection into consideration. In this case study we used participant observation, focus group, interview and analysis of work artifacts (Lethbridge et al., 2005). Data source triangulation uses multiple data sources at potentially different occasions. During this case study, we collected data during multiple sessions with the architects of the project and other stakeholders. Table 17 shows an overview of the sessions, the data collection method used, the data sources, and related research questions.

In the following, the data collection methods used are described in more detail.

- **Participant observation**: Participant observation is a popular data collection method in case studies (Mack et al., 2005; Gray, 2009; Yin, 2003; Seaman, 1999). It involves systematic viewing of actions performed by the observed subjects, recording, analysis and interpretation of the observed behavior. In the observation sessions listed in Table 17, one of the researchers joined the observed subjects during their work. The researcher had access to all documents the subjects used during these sessions and he listened to entire communications in cases where multiple subjects collaborated. During these sessions, the researcher took notes about working topics, time spent, communication issues and other observations that could have a relation to the research questions and their corresponding variables. The session notes and copies of the project artifacts used by the observed subjects were stored in a case study database as proposed by Gray (2009).
- **Focus group**: Focus group data collection is a well-documented technique that assembles small groups of peers to discuss particular topics. Discussion in focus groups is largely open, but it is directed by a moderator allowing soft, or qualitative issues to be explored. Kontio et al. mention additional advantages of focus groups in comparison to other qualitative research methods (Kontio et al., 2004). They observed that the interactive nature of the group discussions with people from different backgrounds encourages participants to react to the comments made by other participants, thus reflecting and building on each other's experiences. It also helps to validate comments and positions, as some points made by participants may result in other participants confirming similar, almost similar or opposite points. These insights would be invisible in personal interviews. The researchers have experience in conducting focus groups from previous studies; therefore a pilot session was not performed. The guidelines presented in Mack et al. (2005) were used to prepare and conduct the focus group. In particular, a question guide was created in advance and internally reviewed by the authors. The questions

Table 17
Data collection methods.

Data collection method	Sessions	Data sources	Research questions
Participant observation	Initial architecture decision elicitation	Architect1, Architect2	RQ1
	Initial creation relationship view	Architect1	RQ1
	Refinement relationship view	Architect2	RQ1
	Discussion about relationship view	Architect1, Architect2	RQ1
	Initial creation of decision detail view, refinement relationship view	Architect1	RQ1
	Discussion of detail view, refinement relationship view, initial creation chronological view	Architect1, Architect2	RQ1
	Planning of architecture review based on decision views	Architect1, Architect2	RQ3
	Revision decision views, revision of architecture review planning	Architect1	RQ1, RQ3
	Architecture review	Architect1, Architect2, three domain experts, two architecture experts	RQ2, RQ3
Focus group	Conducted immediately after the architecture review	Architect1, Architect2, three domain experts, two architecture experts	RQ2, RQ3, RQ4
Interview	Conducted at the end of the case study	Architect1	RQ1–RQ4
	Conducted at the end of the case study	Architect2	RQ1–RQ4
Analysis of work artifacts	n.a.	Existing architecture documentation, produced decision views, outcome of the architecture review	RQ1, RQ2, RQ4

in a question guide are not asked directly, but serve as orientation for the moderator of a focus group. Focus groups are most productive, if the participants are encouraged to have an open discussion, while the moderator tries to lead the discussion in a way that all important questions are answered. The used question guide can be found in Appendix E.

The focus group was conducted by involving all people who took part in a technical architecture review; one of the authors was allowed to join that review. The complete session was audio-recorded with the participants' consent and afterwards transcribed. Additionally, the moderator (researcher) took notes about observations that could not be captured on tape, e.g., collective nodding of participants. The audiotape, notes and the transcript were stored in the case study database.

- **Interview**: Interviews allow researchers to gain in-depth knowledge about the interview topics. Like focus groups, they enable researchers to ask interviewees for clarification to solve potential misunderstandings (Lethbridge et al., 2005). Interviews are an appropriate means to collect opinions and impressions about the object under study (Seaman, 1999). The two interviews with the architects lasted between 50 and 60 min each. They were conducted via videoconferences, because the architects were not on-site in their organization at that time. Both interviews were digitally recorded with the participants' consent and later transcribed. The original audio files and the transcripts were stored in the case study database.
- **Analysis of work artifacts**: This data collection method is used to uncover information about how the architects applied and used decision views by looking at their output. In this particular case, we looked at architecture documentation in wikis, the decision views created by the architects, the architecture review planning document and the architecture review report to gather data relevant for our research questions. All collected documents were stored in the case study database. The authors of the documents were contacted to clarify questions and issues related to the documents.

4.3. Analysis

We use descriptive statistics and qualitative analysis to examine the data gathered during the case study. This section is sub-divided according to the research questions.

4.3.1. Analysis RQ1 – What is the effort of documenting architecture decisions using architecture decision viewpoints?

As described in the study design, the effort of documenting architecture decisions is influenced by some independent variables. Table 18 shows descriptive statistics for the variables defined in Table 10.

The *Duration of the documented project* is the total time in calendar months spent on the project, whereas the *Project size* is the time spent in person-months. The variable *Number of decisions documented* indicates the number of decisions that have been documented using the decision view approach. To the best of our knowledge, the architects documented every design decision they found architecturally significant in the project. The "architectural significance" of decisions, however, is not essential for the decision documentation approach presented here. Many definitions for architecture decisions exist; e.g., architecture decisions are those decisions that have an impact on the system's quality attributes; others emphasize on the external visibility of those decisions. We keep it simple by assuming that a decision is architectural in nature, if the architect believes it is architectural and should be documented.

For the calculation of the average number of words used to document a decision, the words used in every decision detail model were counted; thus no other views were taken into account for this variable.

The effort, according to the dependent variable, is calculated in terms of time spent to create the different views. Table 19 shows the

Table 18
Independent variables RQ1.

Variable	Values
Time the architects have worked in the IT industry	Architect1: 6 years, Architect2: 6 years
Time the architects have worked as software designers/architects	Architect1: 4 years, Architect2: 5 years
Number of architects who created the views	2
Duration of the documented project	13 months
Project size	21 person-months
Number of decisions documented	56
Average number of words used to document one decision in the detail view	61

Table 19
RQ1 – Effort for creating views.

Decision elicitation	Relationship view	Detail view	Chronological view
11 person-hours	4 person-hours	7 person-hours	2 person-hours

Table 20
Percentage effort for creating views.

View	Percentage of total development effort
Relationship view	0.21%
Chronological view	0.15%
Detail view	0.29%

effort in person-hours spent to create views for the corresponding viewpoints. Decision elicitation is shown as an additional category. This is necessary, because architecture decisions have not been documented in the Sandnet project prior to the case study; thus all architecture decisions had to be elicited from existing project documentation and brainstorming sessions by the two architects.

The total effort for creating the three views according to our viewpoint definition was 24 person-hours, which equals 3 person-days and 0.14 person-months. The effort has to be analyzed in the context of the project size. In this case, the project size was 21 person-months, which equals 3696 person-hours (21months × 22days × 8 h). Table 20 shows the effort for creating each view as percentage of the total development effort. In this calculation, the effort for decision elicitation was portioned equally to the three views, as it cannot be assigned to a single view.

We note that in our experience, the order of view creation followed in the case study works well under the assumption that the relationship view is good for an initial documentation of decisions. It is a lightweight view in the sense that decisions can easily be added or removed and related to other decisions. In contrast to the decision detail view, in which decisions have to be described in textual form, revising decisions and relationships in the relationship view is just a matter of drawing ellipses and connectors. This allows for subsequent refinements in quick iterations. Once the main decisions and decision relationships are documented, effort can be invested to capture the decisions in the detail view. Finally, the chronological view adds information about the evolution of the decisions. Remembering the correct chronological sequence of decisions requires a lot of reflection by the architects, if created after the fact. Therefore, in "green field" projects, it can make more sense to create the chronological view right away and revise it iteratively during the project.

The subjects in the case study had to overcome a learning curve, i.e., learn how to use the viewpoints in order to design the views. We assume that the same subjects would be able to document decision views more quickly in future projects. However, this assumption is subject to further empirical evaluation.

The total effort for creating the three views was approximately 0.65% of the total development effort in person-time. In addition to the quantitative analysis of the effort needed to create decision views, we asked the architects about their subjective estimation of the effort. Moreover, we wanted to know which views they would create if they were very limited in time.

The transcripts of the interviews with the architects were analyzed using the *constant comparison method* (Glaser and Strauss, 1967), a well-established theory generation method in qualitative analysis (Seaman, 1999). In detail, the following procedure was followed. The original comments in the interviews were given mainly in colloquial speech and many of them can only be interpreted in the context of the whole interview. We browsed the transcripts of the interviews and searched for passages of text related to the research question. The respective passages were labelled and later grouped into patterns expressing their content as more formal, context-free statements. This procedure was used for the qualitative analysis of the interviews and the focus group conducted after the review. An example of the analysis process is given in Appendix D.

Finally, the original transcript, the extracted comments, and the derived statements were given to two of the participants of the focus group for validation. They concordantly acknowledged that the information in the derived statements is similar to the information in the original comments.

The following statements were derived from text passages in the interviews with the architects, labelled with RQ1:

- How reasonable the effort for creating decision views is, depends on the number of team members and the duration of the project.
- The decision detail view is the most important view for the original architects.
- The decisions in the detail view can be documented with just a few attributes (rather than all of them).
- The decision detail view is not helpful in isolation. It should be complemented at least with a relationship view in order to create an overview of decisions for the other stakeholders who were not directly involved in the decision making process.
- When there is no time to document all views, the architects would skip the chronological view and only create a detail view and a relationship view.
- Important decisions should be documented in more detail than less important decisions.
- The effort for creating decision views is definitely reasonable from the point of view of the project sponsor (the organization funding the project).
- The effort for creating decision views is marginal compared to the whole development effort of the project.

4.3.2. *Analysis RQ2 – Do decision views effectively support stakeholders to understand the architecture?*

As with RQ1, we use descriptive statistics to describe the independent variables. Table 21 shows descriptive statistics for the variables defined in Table 12.

The support for architecture understanding provided by the views is estimated in terms of the level of architecture understanding gained by the stakeholders after studying the views. The analysis of this dependent variable is done qualitatively. During the case study, a technical architecture review was conducted with the architects, a few experts in the network security domain and experts on software architecture. All people, except for the architects, were not familiar with the Sandnet system at all. Two days before the review, all participants were asked to analyze the system by studying the decision views we provided to them. Right after the review, we interviewed all participants in a focus group where we also asked questions about the suitability of the views for understanding the system. The transcript of the audio-recorded focus group is used as a basis for the analysis. We used the constant comparison method, as described in Section 4.3.1. The following lists show the resulting participants' statements for the different viewpoints under study:

Relationship viewpoint:

- Relationship views clearly illustrate relationships between decisions.
- Relationship views support impact analysis.
- Relationship views illustrate decision relationships better than decision detail views.
- Relationship views illustrate dependencies between decisions.

Table 21
Independent variables RQ2.

Variable	Statistics
Time the stakeholders have worked in the IT industry	Stakeholder1: 12 years
	Stakeholder2: 5 years
	Stakeholder3: 15 years
	Stakeholder4: 2 years
Time the stakeholders have worked as software designers/architects	Stakeholder1: 12 years
	Stakeholder2: 5 years
	Stakeholder3: 0 years
	Stakeholder4: 2 years
Time the stakeholder has worked in the network security domain	Stakeholder1: 6 years
	Stakeholder2: 0 years
	Stakeholder3: 14 years
	Stakeholder4: 2 years
How often have the stakeholders been involved in the analysis of architecture decisions	Stakeholder1: Rarely
	Stakeholder2: Frequently
	Stakeholder3: Very rarely
	Stakeholder4: Rarely
Time the stakeholders took to study the views	Stakeholder1: 70 min
	Stakeholder2: 90 min
	Stakeholder3: 35 min
	Stakeholder4: 60 min
Duration of the documented project	13 months
Number of documented decisions	56
Average number of words used to document one decision in the detail view	61

- Relationships views do not contain enough information about a single decision.
- A combination of the decision detail view and the other decision views is necessary.
- The relationship view is good to get an overview of decisions made.
- The relationship view helps to start understanding a system, much better than the decision detail view.

Chronological viewpoint:

- The chronological view helps to understand the evolution of the system.
- Chronological views provide insights into the reasoning process of the architects.
- Chronological views show which solutions were considered, rejected and chosen.
- The chronological view is well suited to understand the decision making process in complex software systems.
- Chronological views are good to analyze change of the architecture over time.

Detail viewpoint:

- Decision detail views are important to analyze the reasoning and the details of every decision.
- A combination of the decision detail view and the other decision views is necessary.
- Detail views are hard to handle in isolation, because they produce large amounts of text on too many pages.

All viewpoints:

- Decision views help to recap the decision making process for a single decision, including considered alternatives.
- Decision views are helpful to communicate architecture decisions to project teams taking over the system.

Table 22
Independent variables RQ3.

Variable	Statistics
Time the reviewers have worked in the IT industry	Stakeholder1: 12 years
	Stakeholder2: 5 years
	Stakeholder3: 15 years
	Stakeholder4: 2 years
Time the reviewers have worked as software designers/architects	*N*: 4
	Stakeholder1: 12 years
	Stakeholder2: 5 years
	Stakeholder3: 0 years
	Stakeholder4: 2 years
Time the reviewers have worked in the network security domain	*N*: 4
	Stakeholder1: 6 years
	Stakeholder2: 0 years
	Stakeholder3: 14 years
	Stakeholder4: 2 years
How often have the reviewers been involved in the analysis of architecture decisions	*N*: 4
	Stakeholder1: Rarely
	Stakeholder2: Frequently
	Stakeholder3: Very rarely
	Stakeholder4: Rarely
How often have the reviewers been involved in architecture reviews	Stakeholder1: 15 times
	Stakeholder2: 5 times
	Stakeholder3: 1 time
	Stakeholder4: 3 times
Duration of the documented project	13 months
Number of decisions documented	56
Average number of words used to document one decision in the detail view	61

- Decision views prevent new project teams or team members from making fatal decisions.
- Decision views help people to understand which decisions were made for which reasons and which decisions were explicitly not made for specific reasons.
- Decision views are very helpful.
- Decision views are a good means to transfer architecture knowledge.
- The architects were amazed how much the stakeholders knew about the system after having studied the views.
- Software engineers should be obliged to create decision views as a complement to the other architecture documentation.
- Decision views capture architectural knowledge that cannot be recovered from traditional views; especially discarded and rejected decisions and the reasoning behind decisions.

4.3.3. *Analysis RQ3 – Do decision views effectively support technical architecture reviews?*

Table 22 contains descriptive statistics for the independent variables described in the study design.

In addition to the variables described in Table 22, the activities performed in the architecture review have an effect as independent variable. In the case study, a technical architecture review was performed based on a custom architecture review method, which will be explained in the following. The review was performed in two phases. In phase one, the reviewers received a review-planning document containing the schedule of the review, a description of the Sandnet project, the main stakeholders, and architecturally relevant requirements, as well as a network topology view and all documented decision views (i.e., a relationship view, a chronological view and a decision detail view). Additionally, they received a description of five technical scenarios. The scenarios had been documented in the company wiki by the architects as possible future changes or enhancements, prior to the case study. An example of a technical scenario as described in the review-planning document is: “Currently, malware samples are executed by a sandpuppet for exactly 1 h. How is the architecture of the system affected if malware samples are executed for a complete day/week or even longer? Currently, there are 80 virtual machine slots for parallel execution of malware samples available”. The reviewers were asked to study the views with respect to the scenarios in advance and write down all uncertainties and questions. Phase one was performed individually and off-site. Phase two was the actual review conducted on-site in the organization. Table 23 shows the schedule of the review and the activities performed.

During the review, the reviewers selected three scenarios out of the five available ones, based on their own judgement of the scenarios’ importance. The detail view and the relationship view were used to identify and analyze decisions that have an effect on the respective scenario. The third selected scenario was skipped because of time constraints. The participants planned an additional review session outside the time-period of the case study.

The support for technical architecture reviews provided by the decision views is estimated in terms of the number of risks that came up during the review and the level of support for the performed reviewing activities. To estimate the number of identified risks during the architecture review, we analyzed the risk evaluation forms filled in by the reviewers during the scenario-based reviews. Risk evaluation forms are part of the Software Risk Evaluation (SRE) Method (Williams et al., 1991) defined by the Software Engineering Institute, which was used by the architects to evaluate the risks uncovered during the review. In total, the four reviewers recorded 27 distinct risks (reviewer 1: 3 risks, reviewer 2: 4 risks, reviewer 3: 12 risks, reviewer 4: 8 risks). Out of the 27 risks, the architects regarded five risks as high or medium severity. The other risks were either low severity, or the architects did not share the reviewer’s opinion. The analysis of the support for reviewing activities was done based on an examination of the focus group transcripts, in which the support for architecture reviews was explicitly discussed. For the qualitative analysis we followed the same procedure as described for RQ1. The following lists show the derived participants’ statements assigned to the respective reviewing activities:

Activity: Architect explains evolution of the system using the chronological view

- The chronological view helps the architect to explain the evolution of the system.
- The chronological view helps to explain and remember the change of decisions over time.
- The chronological view helps reviewers to understand the evolution of the system.

Activity: Architect clarifies questions with respect to the system

- Decision views are well suited for explaining the architecture to stakeholders.
- Decision views can make sure that nothing is forgotten when explaining the architecture.
- Decision views make sure that the architecture can be communicated in a structured and understandable way.

Activity: Review scenarios

- The logical groups in the relationship view help to structure the decisions.
- The logical groups in the relationship view help to keep the overview over decisions.
- The relationship view supports the architect to perform impact analyses.
- The chronological view can be used to analyze changes between architecture iterations.
- The chronological view allows looking up changes between iterations quickly.
- Relationship views help reviewers to identify critical issues in the architecture.
- Relationship views visualize decision alternatives and allow reviewers to evaluate the final choice among the alternatives.
- Without decision views, the decisions and their relationships have to be elicited during the review.
- Decision views are a good basis for architecture reviews.
- Relationship views emphasize central decisions and decision alternatives.
- Relationship views allow identifying critical decisions quickly.
- Decision views capture architectural knowledge that cannot be recovered from traditional views. Especially discarded and rejected decisions and the reasoning behind decisions.
- The chronological view helps to understand changes between architectural milestones.
- Decision views allow reviewers to reassess if the architects have evaluated the right decision alternatives soundly.
- Decisions views can be used to assess if architectural problems were analyzed correctly.
- Decision views are helpful as a complement to traditional architecture documentation.

4.3.4. Analysis RQ4 – Do decision views support architects to distill reusable decision sub-graphs?

As for the fourth research question, we present descriptive statistics for the independent variables. Table 24 shows the results.

The support that decision views provide for distilling reusable decision sub-graphs is estimated in terms of the number of concrete reusable decision sub-graphs identified by the stakeholders and architects; and the level of support the decision views provide

Table 23
Review schedule.

Time	Activity
14:10	Start
14:20 to 14:30	Introduction of the Sandnet project by the architects
14:30 to 14:40	Introduction of the review process and goals by the review organizer
14:40 to 14:55	One of the architects explained the evolution of the system using the chronological view
14:55 to 15:05	Choice of three scenarios out of the five scenarios described in the review-planning document
15:05 to 15:50	Review scenario 1 using the decision detail view and the relationship view
15:50 to 16:30	Review scenario 2 using the decision detail view and the relationship view
16:30 to 17:00	Wrap-up session including discussion and documentation of all discovered issues

Table 24
Independent variables RQ4.

Variable	Statistics
Time the stakeholders have worked in the IT industry	Stakeholder1: 12 years
	Stakeholder2: 5 years
	Stakeholder3: 15 years
	Stakeholder4: 2 years
Time the stakeholders have worked as software designers/architects	Stakeholder1: 12 years
	Stakeholder2: 5 years
	Stakeholder3: 0 years
	Stakeholder4: 2 years
Number of decisions documented	56

for identifying reusable decision sub-graphs. The latter is analyzed based on the transcripts created from the focus group after the review, in which we asked the participants to identify reusable decisions paths with the help of the documented views and the transcripts from the interviews with the architects.

The architects and stakeholders found the relationship view helpful to identify reusable decision sub-graphs. The relationship view documented for the Sandnet project contains six logical decision groups with 56 decisions in total. The stakeholders and architects were asked to identify concrete decision sub-graphs that could be reused in other software projects. From the six logical groups, they selected four groups that contain reusable decisions. Group one contains decisions about a web application used to access analysis data and configure sandpuppets; group two contains decisions related to the used database management system; the decisions in group three are about the control mechanism for the herders; and group four contains decisions related to the virtual machine technology and configuration used. The decisions in the remaining two logical groups were regarded as being too specific for being reusable. The analysis of the level of support was done qualitatively—similarly to the qualitative analysis described in RQ2 and RQ3. The first set of statements was given by the architects during the interviews, the second set of statements was made by the other stakeholders during the focus group after the review.

Architects:

- When identifying reusable decision sub-graphs, the dependencies upon other decisions can be evaluated using the relationship view.
- The relationship view provides good support for identifying reusable decision sub-graphs.
- Decisions that are strongly coupled to concrete requirements cannot be reused.

Stakeholders:

- Decision views are helpful to derive reusable decision sub-graphs.
- Logical decision groups in the relationship view are strong candidates for reusable paths.
- Logical decision groups in the relationship view are optimal decision combinations in the given context.
- The relationship view is well suited for extracting reusable architectural knowledge.
- Relationship views help to speed up the start of new projects.
- Relationship views provide guidance for architects with respect to dependencies between decisions and decision options.
- When reusing decisions from the relationship view, it is important to consider the context in which the decisions were made.
- Decision views help architects to recap the decisions made.
- For reusing decisions from the relationship view, the decision detail view is needed to judge if the context and the requirements behind the decision match.
- Logical decision groups in the relationship view are strong candidates for reusable paths.
- It is important to look at the goals of the decisions before reusing them.
- Decisions can be directly reused if the architectural goals behind the decisions are compatible with the goals of the new project.
- If the decisions' goals of the new project are not compatible with those of the documented project, then the decision views can still be used to identify candidate decisions that have to be reevaluated in other projects.
- Decision views provide reusable decision alternatives.
- Decision views allow reusing decisions for comparable problems.
- Decision views provide a basis for decision-making processes in other projects.
- By studying decision views for reuse, architects can make sure they do not forget decision options.

4.4. *Interpretation*

In this section, the results of the analysis are interpreted. At the end of the section, we discuss potential threats to the validity of this study.

4.4.1. *Interpretation RQ1*

Research question one is about the effort needed to create decision views according to our viewpoint specifications. In the analysis, we showed that the documentation of the decision views took approximately 0.65% of the total development effort of the software project under study. For the Sandnet project, this means that less than 1 h (39 min) out of 100 person-hours had to be spent in order to create a relationship view, a chronological view and a decision detail view. The effort needed to create an individual view is hard to generalize, as it depends on the order in which the views were created. In this case study, the architects had not documented architecture decisions at all prior to the case study; thus all decisions had to be elicited in the first place. This took by far the greatest effort. Creating the detail view takes the second greatest effort. Finally, the relationship view and the chronological view follow in the effort ranking. Once decisions are documented in the detail view, the effort for creating the other views for the decisions is relatively low.

From the interviews with the architects, we learned that they would document the detail view first, but they would not describe every decision at the same level of detail. Some decisions might be more important or more complicated than other decisions; these decisions should be described in detail. Other decisions that are easy to comprehend do not have to be documented in detail. After the detail view, the architects would create a relationship view. The effort for creating the chronological view—at least in the opinion of the Sandnet architects—only pays off if projects are subject to long-term evolution.

It can be concluded that the effort for creating decision views is relatively low compared to the complete development effort. Depending on the available time, architects may choose not to document all views, but choose a subset depending on the characteristics of the concrete project. The architects mutually agreed that the effort for creating decision views is reasonable from the point of view of the project sponsors, which means the cost-benefit ratio of the decision views is low.

4.4.2. *Interpretation RQ2*

Research question two concerns the support for stakeholder communication offered by the views. The independent variables presented in the analysis show that the people who used the views to study the architecture of the Sandnet project were rather inexperienced with respect to architecture decision analysis. On average, they stated that they were rarely involved in the analysis of architecture decisions and that they have worked in the IT industry for less than ten years, which means that they are rather inexperienced technical stakeholders. On average, they took slightly more than 1 h to study the decision views of the Sandnet project; none of them knew the project in advance. It is notable that the architects of the Sandnet project were astonished by the knowledge the stakeholders gained about the project just by analyzing decision views. This impression is supported by the fact that the short preparation time was sufficient to identify major risks during the architecture review. The stakeholders concordantly stated that the relationship view is well suited to get an overview of the decisions made and to understand the relationships and dependencies between the decisions. The chronological view helped them to understand the evolution of the system and to get an insight into the architects' reasoning process. The decision detail view was seen as an important complement to the other two views in order to grasp the rationale behind a single decision. Finally, all stakeholders and the architects of the project agreed that decision views are a good means to communicate architecture decisions.

4.4.3. Interpretation RQ3

RQ3 is about the suitability of decision views to support architecture reviews. With one exception, the reviewers were rather inexperienced in performing architecture reviews. Nevertheless, the analysis showed that they were able to find 27 risks in the current architecture, from which the architects confirmed five risks as being important.

Decision views are beneficial for the preparation of architecture reviews, as they give an overview over decisions made, show dependencies between decisions and allow comprehending the rationale behind single decisions. The chronological view allows reviewers to recap the evolution of the system and to quickly identify decisions that have changed since the last architectural review. The relationship view was found especially well suited for identifying important and critical decisions, performing impact analyses, and finding dependencies between decisions. The fact that rejected decisions and discarded decision alternatives are shown in the views allows the evaluation of single decisions quickly. Moreover, decision views support the architects in communicating the architecture to the reviewers in a structured way, thus ensuring that no important decisions are forgotten. The participants of the review mutually agreed that decision views are a good basis for architecture reviews that supports many review activities; particularly the introduction of the architecture to the stakeholders and the discussion of review scenarios.

4.4.4. Interpretation RQ4

Decisions views offer support for identifying reusable decision sub-graphs. Especially the relationship view was found helpful for identifying reusable decisions. The logical decision groups within this view are candidates for decision sets that can be reused as a whole. However, the reusability always depends on the context, in which decisions were made. Decisions from other projects can only be reused if the context and the architecturally significant requirements are comparable. The information from the decision detail view is essential to judge this for single decisions. One important aspect of decision views is that they record decision alternatives. This information is often not part of an architecture description, because many alternatives have not made it in the final architecture. Decision alternatives constitute valuable information for decision sub-graphs as they allow one to prepare decisions for question-option-criteria trees (MacLean et al., 1991), i.e., if requirement A then decision alternative B, if requirement C then decision alternative D and so on. Finally, studying decision views from projects in the same domain can help architects to make sure that no important decisions or decision alternatives have been forgotten.

4.4.5. Threats to validity

Construct validity is the degree to which the case is relevant with respect to the research questions (Runeson and Hoest, 2009). A frequent problem in case study research is that the case study design fails to clearly define operational measures, which allow one to objectively judge the collected data (Yin, 2003). In this case study, we clearly defined the research questions prior to the data collection phase. The methods for the data collection were systematically selected in order to sufficiently address all four research questions. Furthermore, for every research question, we defined the dependent variables used as "measurements" to judge the data as well as the independent variables influencing those measurements.

We used different means to improve the internal validity of our findings. Internal validity concerns hidden factors, which affect the dependent variables (Wohlin et al., 2003). Using different types of triangulation can increase the reliability of the study results (Runeson and Hoest, 2009; Lethbridge et al., 2005). In this case study, we used several types of data source triangulation, e.g., by performing interviews with different people or by looking into different work artifacts. We also used methodological triangulation by combining different types of data collection methods. With the two types of triangulation, we made sure that every research question was addressed by more than one data source and by using different collection methods.

A potential threat to internal validity is the identification of the technical scenarios evaluated in the architecture review. The architects who defined the scenarios could have consciously or unconsciously defined scenarios that are well supported by the decision views. This could have resulted in a higher valuation of the decision views by the review team. This, however, was not the case. The scenarios were defined by the architects prior to and independent from the case study. They were documented as potential future changes or enhancements in the company wiki. To partially eliminate bias on the side of the architects, the review team chose three out of the five scenarios based on their own estimation of the scenarios' importance.

External validity is related to the generalizability of the results with respect to a specific population. External validity is regarded as a major problem in case study research because only one case is studied; which makes statistical generalization impossible (Yin, 2003). We believe that our findings are valid at least for projects that have a comparable size (in person-months and development team) and use similar architecture approaches. However, external validity can only be shown (e.g., by analytic generalization, Yin, 2003) with at least a few replications. Although we have observed many of our findings in the three pilot projects (see Section 4.2) before conducting the case study, a systematic replication of the study in different projects and organizations is needed to support the claim for external validity.

5. Related work

Our work is related to the following fields within software architecture: architecture decision documentation and architecture decision views.

5.1. Decision documentation approaches

There are currently three main approaches to documenting architecture decisions. We briefly present these approaches and describe how our proposed viewpoints relate to each of them:

Decision templates: Different templates have been proposed to describe architecture decisions in textual form, mainly using tables (see for instance Tyree and Akerman, 2005). They can be used to capture relevant rationale behind decisions including, among others, assumptions, alternative decision outcomes, the decision state, related requirements, and possibly related decisions. Tabular decision descriptions offer a certain degree of freedom for the decision documenters, because description elements can easily be added or left out. An additional benefit is their suitability for simple automated support such as through spreadsheets or wiki-type information systems, which offer out-of-the-box support for creating tables and linking textual elements to each other. No extra notations or special tools are needed to document decisions in textual form. However, decision tables tend to become very large and contain a lot of text. When many decisions are documented for one system, the overview of these decisions gets lost. Using templates, it is also challenging to visualize or trace complex relationships between decisions and to perform impact analyses. Especially for non-technical stakeholders and managers, long architecture documentation may seem daunting, which discourages them from reading the documentation.The details captured using decision templates are of paramount importance for framing a number of concerns. We have thus decided to include a viewpoint, the Decision Detail viewpoint, that uses a decision template similar to Tyree and Akerman (2005). The views resulting from applying this viewpoint are especially important to record rationale for decisions. However, as mentioned before, they are not sufficient. Additional views are needed to provide an overview of the decisions made and to emphasize concerns that can hardly be satisfied in a table or catalog, e.g., decision relationships, decision chronology and the impact of decisions on stakeholders, on the architecture or on other decisions.

Annotations: Other approaches document architecture decisions using annotations (see for instance the Knowledge Architect Suite, Liang et al., 2009). The respective tools allow users to attach "comments" to other architecture descriptions such as to UML or ADL models or to natural language text in text processing applications. They highlight elements as decisions and additionally capture relations, attributes, and the history of decisions. An advantage of using annotations is that architects and other stakeholders need not learn new tools to document decisions. Different stakeholders can typically use their preferred tools and attach annotations through specialized plugins to those tools. Furthermore, annotations are very well suited to elicit architecture decisions from existing project documentation. On the other hand, annotations from different tools must be combined with an additional decision management or documentation approach that allows one to consume decisions without browsing multiple documents using multiple tools. Additionally, annotations do not provide instant support for the analysis of decision relationships, lifecycle management and consistency checks. This must be accomplished by additional tools collecting the annotated information and preparing them for further analysis (e.g., the Microsoft Word plugin used in Liang et al., 2009). The elicited decisions from an annotation approach can serve as the raw data basis for creating decision views conforming to architecture decision viewpoints. Therefore annotation approaches are complementary to the proposed decisions viewpoints.

Decision models: Decision models can present the same information as template-based and annotation-based approaches, but use dedicated models to represent decisions. Existing models within an architectural view are complemented with a decision model, which addresses decision-related concerns specific to the particular view (Kruchten et al., 2009; Dueñas and Capilla, 2005; Capilla et al., 2007). For example, in an architecture description following the 4+1 view model, a decision model in the deployment view would contain decisions about system deployment.This approach provides a better overview of decisions than the aforementioned approaches and facilitates linking decisions to other architecture description elements. However, the problem remains that the existing approaches on decision models are dedicated to existing architectural viewpoints. The concerns addressed by the models are the same (or a subset of) concerns that are addressed by the viewpoint they are a part of; thus are system concerns. We, however, argue that beyond system concerns, architecture decision documentation must address additional concerns specific to architecture decisions. Nevertheless, architecture decision models can be complemented with our decision views if appropriate means for consistency among the decisions are taken. Additionally, the decisions from the decisions models can serve as a basis for further architecture decision elicitation.

5.2. Architecture decision views

The idea of introducing a dedicated decision view[3] to complement the traditional architectural views (e.g., the 4+1 view model Kruchten, 1995) has been proposed by Kruchten, Capilla and Dueñas (Kruchten et al., 2009; Dueñas and Capilla, 2005). They emphasize the importance of documenting design rationale as a part of architecture decision documentation in architecture practice and identify a set of challenges and benefits of decision documentation. However, no concrete guidance is provided on how to define and construct decision views that integrate with view-based architecture documentation (ISO, 2011). The documentation framework presented in this paper addresses many of the challenges identified in Kruchten et al. (2009) and gives concrete advice on how to construct a set of consistent architecture decision views.

Various authors have proposed tools to visualize architectural design decisions (for a comparison of current toolsets, see Shahin et al., 2010). Graphical representations of decisions support stakeholders in understanding the architecture, as they allow them to visually inspect the architecture (Shahin et al., 2010; Lee and Kruchten, 2008).

Some tools allow visualizing architecture decisions from different perspectives. A *perspective*, in this context, is a graphical representation of decisions suitable to address a set of decision-related concerns. While the idea of showing different perspectives of decisions has commonalities with the concept of multiple architecture decision viewpoints, it is not sufficient in isolation. A viewpoint is more than a perspective on architecture decisions, as it must provide concrete guidance on how to construct views, and it must ensure inter-model and inter-view consistency. Most importantly, it must integrate with other views used to document architecture. Our work is complementary to the tools analyzed in Shahin et al. (2010); in fact, the creation of multiple views on decisions is only feasible in practice if appropriate tooling is provided to support the architects. This became evident in our case study. Identifying the best technique used to visualize the views presented in this paper is, however, subject to further research. Candidate techniques should adhere to viewpoint definitions and be validated in industrial case studies.

[3] An early draft of IEEE 1471 (version D1.0, dated February 1998) contained a decision viewpoint, described as: "The decision viewpoint documents the decisions about the selection of elements or characteristics. This viewpoint records the rationale for architectural choices. Typical models include: mission utility, cost/capability tradeoffs, element performance tradeoffs". However, all predefined viewpoints were removed from the standard before the final publication, leaving definition of viewpoints to its end users.

6. Conclusions and future work

In this paper, we introduced a documentation framework for architecture decisions consisting of four initial viewpoint definitions and the respective correspondence rules to ensure consistency among them. The four viewpoints, a Decision Detail viewpoint, a Decision Relationship viewpoint, a Decision Chronology viewpoint and a Decision Stakeholder Involvement viewpoint, satisfy several stakeholder concerns related to architecture decision management. Furthermore, they can easily be integrated with other viewpoints to complete the picture of architectural design, decisions and rationale.

With the exception of the Decision Stakeholder Involvement viewpoint, we validated the framework in an industrial case study and showed that the views can be created with reasonable effort. Furthermore, we showed that decision views facilitate communication between stakeholders, support technical architecture reviews and enable the reuse of architecture decisions. Although we could only find evidence for the suitability for *technical* architecture reviews in the case study, we believe that decision viewpoints are equally beneficial for non-technical architecture reviews and evaluations. We are currently planning a study to provide empirical evidence for this assumption.

The framework presented in this paper comprises a coherent set of viewpoints that can be used as-is to document architecture decisions. However, our analysis of stakeholder concerns related to decision documentation (see Section 2) showed that some concerns cannot be satisfied optimally within the current set of viewpoints. In particular, concerns related to decision-requirements traceability and decision-design traceability (C6, C7, C12) in Table 1 are currently under-represented and require additional research. To partially bridge this gap, we are working on an additional viewpoint focussing on the representation of the relationships between architecture decisions and system concerns. In this so-called *decision forces viewpoint*, we represent decisions and the forces that had an influence on the decision-making process. These forces include traditional functional and non-functional requirements, but they also take the experience and expertise of the development team, as well as business and projects constraints into account.

Another direction for future research is applying the documentation framework for creating views in different orders and at different levels of detail. When applying our viewpoints in different projects and different organizational contexts, we observed that the viewpoints can be used in different ways. To give an example, in one project the decision detail view of architecture decisions was created on-the-fly during the decision making itself. This resulted in all decisions being documented with the same effort and the same information density. The decision relationship view was created after-the-fact at the end of the architecture phase. In another project, the relationship and chronological views were created on-the-fly during the architecting process, while only the most important decisions were thoroughly documented in the detail view afterwards. We plan to analyze the different ways of using the viewpoints in more industrial projects to come up with parameterized guidelines on how to construct the views in different organizational settings.

Finally, as mentioned before, effective tooling is vital for using viewpoints in the industry. We developed an open source web application (Open Decision Repository) to create views according to this viewpoint framework. Currently, the Decision Detail, Decision Relationship and Decision Chronology viewpoints are supported by the tool. The source code and documentation is located in a Google code repository and can be found under http://opendecisionrepository.googlecode.com. We are currently evaluating the Open Decision Repository in an industrial study as a pilot for larger-scale empirical validation.

Acknowledgements

We thank the following people for their valuable input and contribution: Stefan Arians, Dirk Bugzel, Mathias Deml, Christian Dietrich, Veli-Pekka Eloranta, Matthias Galster, Kai Koskimies, Christian Manteuffel, Dominique Petersen, Ben Ripkens, Christian Rossow, Sebastian Schmidt, Michael Stal, Martin Verspai, and the participants of the OPR software factory 2011.

We would also like to thank the anonymous reviewers of JSS, whose comments on this paper helped us to improve it.

Appendix A. Concern analysis

Table A.1 shows the outcome of the concern analysis described in Section 2.

Each row contains an architectural knowledge management (AKM) use case elicited from the literature (Liang et al., 2009; Kruchten et al., 2006; Jansen et al., 2007), the concerns derived from these use cases (please refer to Table 1 for a description of the concerns), and typical stakeholders having those concerns for architecture documentation. Additionally, the table indicates how the concerns were derived (column *DER*). The table is ordered by the publications, from which the use cases were elicited.

The following activities were used to derive concerns:

- **Derive (DER)**: A concern or a set of concerns was derived from a decision-related use case.
- **Project (PRJ)**: A use case that does not directly involve architectural decisions was projected to architectural decision concerns.
- **Complement (COM)**: A new concern was introduced to complement concerns derived from a use case.

The analysis was done by the three authors. In cases where the three authors identified different concerns, a discussion took place to reach consensus.

Appendix B. Decision views from the case study

See Figs. B.1–B.3

Appendix C. Viewpoint definitions and correspondence rules

C.1. Decision framework metamodel

Fig. C.1 shows a shared metamodel for the decision viewpoint elements. The metamodel is not specific to one particular viewpoint; instead it is common to all decision viewpoints introduced in this paper. Elements with a gray background map to the corresponding elements in Figs. 2 and 4 of ISO/IEC/IEEE 42010 (ISO, 2011). Therefore, the architecture description elements used in the architecture decision viewpoints (white background) integrate seamlessly into the conceptual framework of the standard.

A shared metamodel, together with well-defined constraints and correspondence rules, can ensure consistency among the views from different viewpoints. The intra-model constraints will be defined later, as a part of the viewpoint definitions. Additionally, inter-model and inter-view correspondence rules will be defined then to ensure consistency between the views.

C.2. Decision Relationship viewpoint

As mentioned in Section 3, the relationship viewpoint describes relationships between architectural design decisions. Table 4

Table A.1
Decision concerns derived from use cases.

Use case	Derived concerns	DER	Typical stakeholders
If we want to do a change in an element, what are the elements impacted (decisions, and elements of design) (Kruchten et al., 2006)	C13	PRJ	Architects
Find out if multiple systems can be combined (migrated) (Kruchten et al., 2006)	C15, C14, C9, C8	PRJ, COM	Architects, Reviewers
From a given perspective (such as security, safety, reuse, etc.) what are the knowledge elements involved? (Kruchten et al., 2006)	C6	PRJ	Architects, Reviewers, Customers, Requirements engineers
You want to integrate multiple systems and decide whether they fit. The tool would help answering questions about integration strategies (Kruchten et al., 2006)	C15, C14, C9, C8	PRJ, COM	Architects, Reviewers
What pieces of Architectural Knowledge have been added or modified since the last review? (Kruchten et al., 2006)	C21, C22	PRJ, COM	Architects, Reviewers
The architect makes sure that all the dependencies of removed AK (i.e., the consequences of an architectural decision) have been removed as well (Kruchten et al., 2006)	C11, C22, C8	PRJ, COM	Architects, Reviewers
What pieces of Architectural Knowledge have been added or modified since the last review? (Kruchten et al., 2006)	C21	PRJ	Architects, Reviewers
Over a time line, find what the sequence of design decisions has been (Kruchten et al., 2006)	C20	DER	Architects, Reviewers, New Project Members
Identify decisions being hubs (god decisions) (Kruchten et al., 2006)	C10, C12	DER, COM	Architects, Reviewers
Identify circular dependencies (Kruchten et al., 2006)	C10	DER	Architects, Reviewers
Identify decisions that gain weight over time and are more difficult to change or remove (Kruchten et al., 2006)	C10, C21	DER, COM	Architects, Reviewers
Identify the stakeholder who seems to have the most “weight” on the decisions, and who therefore maybe the one that could be most affected by the future evolution of the system (Kruchten et al., 2006)	C16, C18, C19, C17	DER, COM	Architects, Reviewers, Manager
Identify who are the stakeholders whose changes of mind are doing the most damage to the system (Kruchten et al., 2006)	C18, C21, C22	PRJ, COM	Architects, Reviewers, Managers
Identify patterns in the decision graphs that can be a useful fashion and lead to guidelines for the architects (Kruchten et al., 2006)	C23, C5	DER, COM	Architects, Reviewers, Domain Experts
Trace between various AK elements, e.g. design decisions, rationale, and design (Liang and Avgeriou, 2009)	C11, C6, C4, C3, C19, C5, C17	PRJ, COM	Architects, Reviewers, Customers, Requirements engineers, New Project Members
The reviewer performs a critical evaluation of the AK, e.g. to make sure that requirements have been satisfied in the architecture design (Liang and Avgeriou, 2009)	C6, C7, C4, C3, C2	PRJ, COM	Architects Reviewers, Customer, New Project Members
Perform an evaluation of architectural knowledge (Liang and Avgeriou, 2009)	C4, C3, C5	PRJ, COM	Architects, Reviewers, Customers, New Project Members
The architect evaluates when the architecture can be considered as finished, complete, and consistent, e.g. verify whether a system conforming to the architecture can be made or bought (Liang and Avgeriou, 2009)	C6, C7, C9, C8, C2	PRJ, COM	Architects, Reviewers
Browse architectural knowledge dependencies (Liang and Avgeriou, 2009)	C10	PRJ	Architects, Reviewers
Browse architectural knowledge traces (Liang and Avgeriou, 2009)	C11, C16, C12, C6, C4, C8, C19, C5, C17	PRJ, COM	Architects, Reviewers, Customers, Managers
Understand the rationale of a design decision (Liang and Avgeriou, 2009)	C3, C5	DER, COM	Architects, Reviewers, Customers, New Project Members
Distill specific knowledge from a system into general knowledge (e.g. architecture pattern) that can be reused in future systems (Liang and Avgeriou, 2009)	C23, C5	DER, COM	Architects, Reviewers, Domain Experts
Produce a consistent subset of Architectural Knowledge to prime the pump for a new system (reuse Architectural Knowledge) (Liang and Avgeriou, 2009)	C23, C5	DER, COM	Architects, Reviewers, Domain Experts

shows the concerns framed by the viewpoint as related to the mentioned stakeholders.

C.2.1. *Model kind*

Fig. C.2 shows a metamodel for the relationship viewpoint. It documents the model kind, which presents the conceptual elements for architecture models that adhere to it. It uses the notation for class diagrams from the Unified Modeling Language. One relationship view can contain multiple relationship models of this model kind; however, every decision is represented only once in a view.

An architecture decision is identified by a short name. Although an architecture decision has potentially many versions, one for every state change, the relationship view contains only the current versions of the decisions shown. A decision has a state, which can be freely chosen depending on the needs of the respective development project. All possible states must be clearly specified prior to being used. With one exception, we adopt the decision states from Kruchten's ontology of architectural design decisions (Kruchten, 2004):

- **Idea**: This state is used for decisions which are just loose ideas that architects want to document so that they do not get lost. If a decision has the idea-state, then it cannot have any relationships to other decisions.

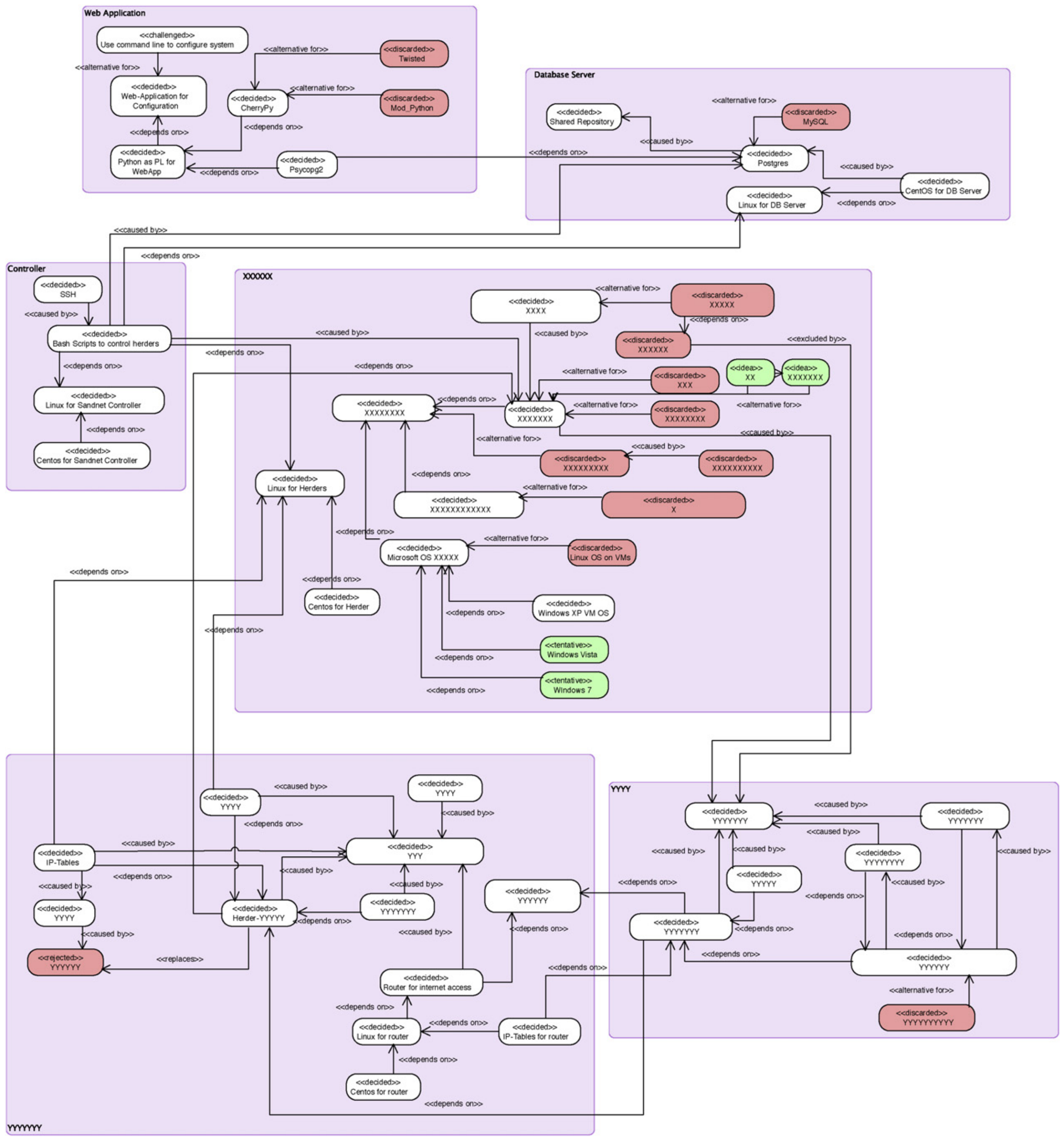


Fig. B.1. Partially censored excerpt from a relationship view.

- **Tentative**: This state is used for decisions which are seriously considered by the architect.
- **Decided**: The decision reflects the current position of the architect and must be consistent with other decided decisions.
- **Approved**: This state is reached if a previously decided decision has been confirmed; for instance during a review or a customer meeting.
- **Challenged**: This state is applicable, if a stakeholder raises issues about a previously decided or approved decision.
- **Rejected**: A rejected decision is a decision that was challenged and has been removed from the current iteration of the architecture. For the sake of simplicity, we subsume Kruchten's **Obsolesced**-state under this state as well.

In addition to Kruchten's states, we define the state *discarded*. A discarded decision is a formerly tentative decision that was not decided, for instance a design option that was not chosen among the considered alternatives. Fig. C.3 shows the decision states along with the respective state transitions.

Decisions participate in relationships. Every relationship refers to exactly two decisions, one source and one target decision. For instance, decision1 (source) replaces decision2 (target). For the sake of simplicity, the meta model only takes binary relationships into account, although in some cases n-ary relationships between decisions may be useful.

A relationship has a specific relationship type, which again can be freely chosen, but should be clearly specified. We define the following relationship types:

- **Depends on**: If decision B depends on decision A, then B cannot be decided or approved without A being in that state. Expressed the other way round, A is a prerequisite for B.

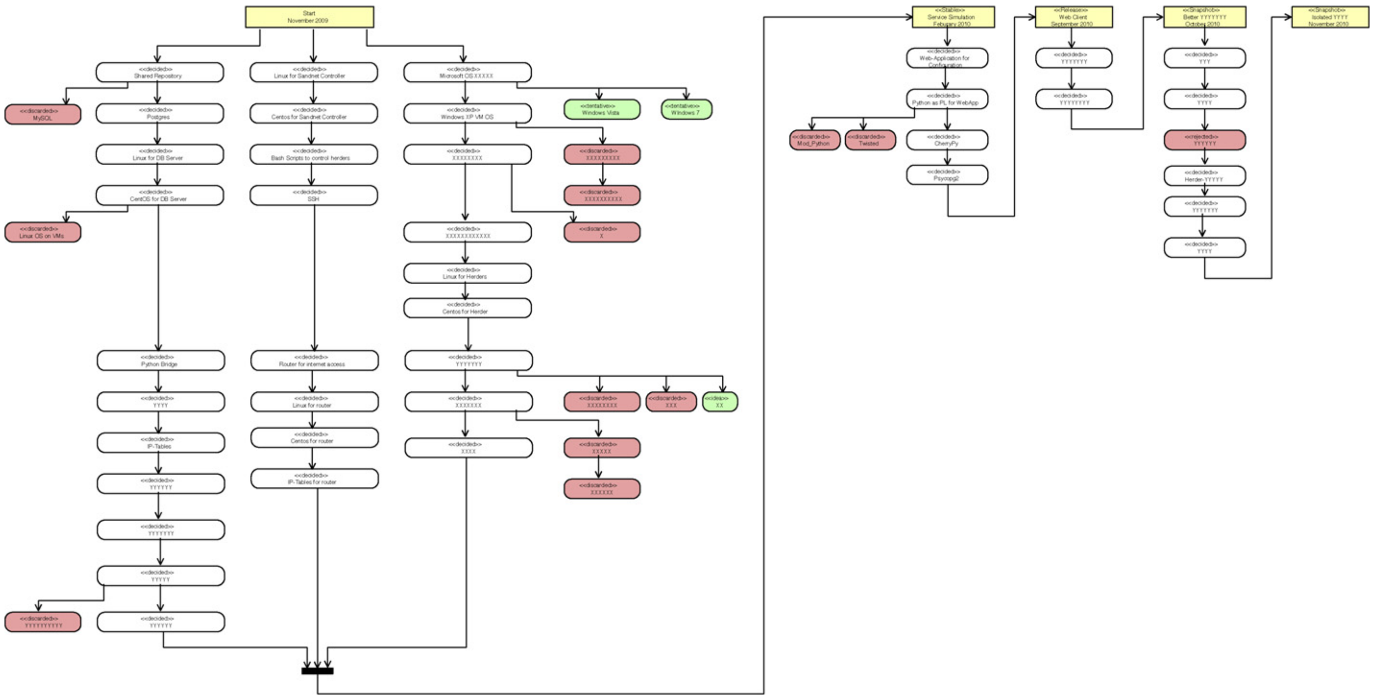
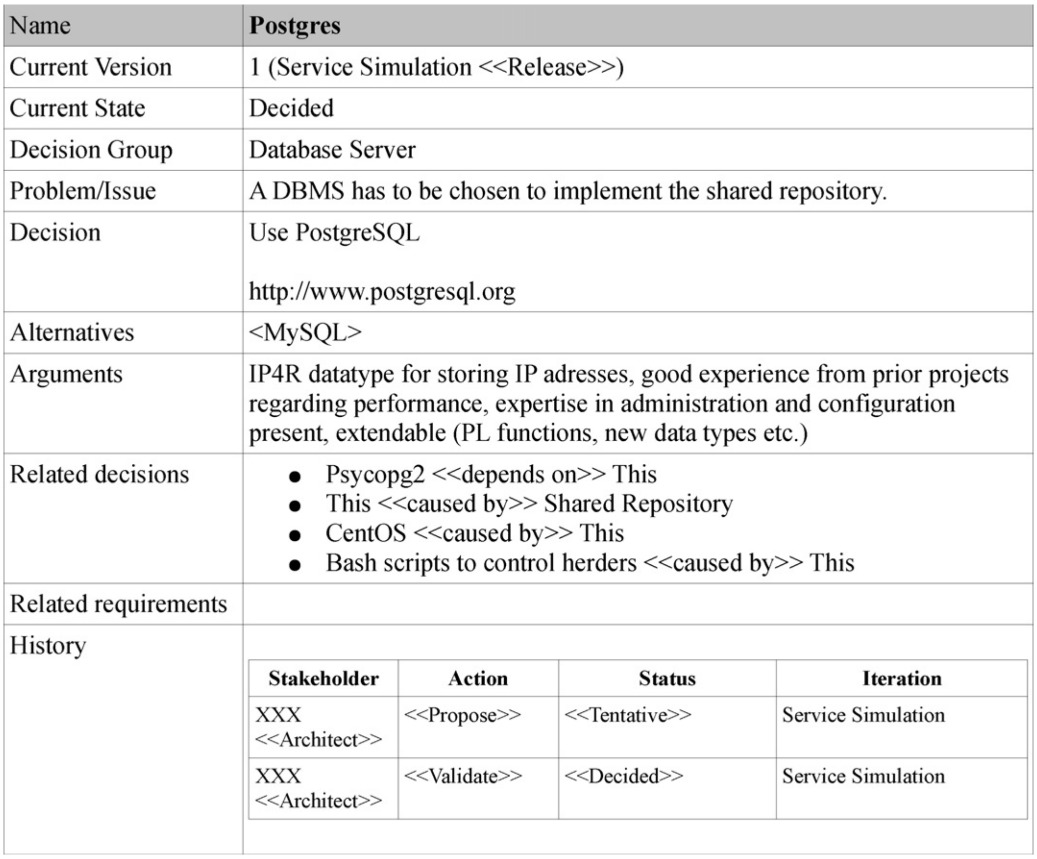


Fig. B.2. Partially censored excerpt from a chronological view.

This relationship includes Kruchten's cases in which decision B is part of a decomposition of A, or if B is comprised by A.

- **Caused by**: If decision B is caused by decision A, then B would not have been decided without A being decided. This relationship expresses causality, without imposing further constraints on the decisions.
- **Is excluded by**: Decision A is excluded by decision B if A cannot be decided as long as decision B is decided. In other words decision B prevents decision A.



Name	**Postgres**
Current Version	1 (Service Simulation <<Release>>)
Current State	Decided
Decision Group	Database Server
Problem/Issue	A DBMS has to be chosen to implement the shared repository.
Decision	Use PostgreSQL http://www.postgresql.org
Alternatives	<MySQL>
Arguments	IP4R datatype for storing IP adresses, good experience from prior projects regarding performance, expertise in administration and configuration present, extendable (PL functions, new data types etc.)
Related decisions	• Psycopg2 <<depends on>> This • This <<caused by>> Shared Repository • CentOS <<caused by>> This • Bash scripts to control herders <<caused by>> This
Related requirements	
History	

Stakeholder	Action	Status	Iteration
XXX <<Architect>>	<<Propose>>	<<Tentative>>	Service Simulation
XXX <<Architect>>	<<Validate>>	<<Decided>>	Service Simulation

Fig. B.3. A single model (decision) in the detail view from the case study.

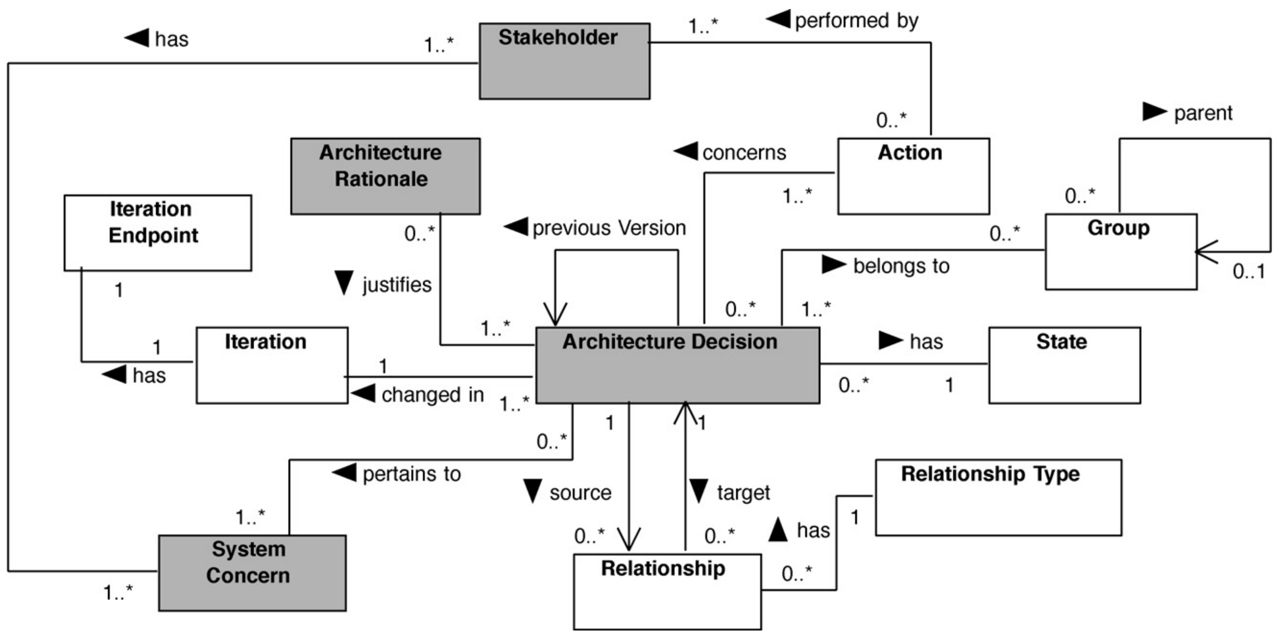


Fig. C.1. Metamodel of decision viewpoints elements.

- **Replaces**: Decision B replaces decision A, if B was put into practice instead of A.
- **Is alternative for**: If decision B is an alternative for decision A, then B was considered as design option instead of A. Two decisions are alternatives when they address a significant common set of concerns.

Table C.1 shows a mapping of our types to the relationship types defined in Kruchten's ontology (Kruchten, 2004). An arrow after the relationship type name indicates that the relationship types in that row are complementary—they may be expressed in either of two ways: e.g., if decision A was *caused by* decision B, then decision B *enables* decision A.

A decision can belong to zero or more decision groups. This allows for logical grouping of decisions according to self-defined characteristics. For instance, decisions could be grouped by subsystem, use-case package, physical location, or component. Decision groups can have parent groups. This is especially helpful to organize the documentation of large numbers of decisions. The models in the relationship view can provide different "scales", e.g., one model showing only the root decision groups and their relationships and additional models for "zooming into" each of the groups showing either decisions or subgroups, which themselves contain decisions or further subgroups.

The following constraints apply to the elements within this model kind:

1. The architecture decisions shown in one relationship view all refer to the same point in time.
2. Every decision occurs exactly once.
3. A decision has a unique name and exactly one state.
4. A decision can participate in zero or more relationships.
5. A relationship has exactly one type.
6. A relationship has exactly two non-identical endpoints.
7. A relationship model showing decision groups without associated decisions must be refined by one or more additional relationship models showing which decisions belong to which decision group.
8. *Caused by*-relationships cannot point to *idea* or *discarded* decisions.
9. *Caused by*-relationships cannot originate from *idea* decisions.
10. *Depends on*-relationships can only point to *tentative*, *decided*, *approved* or *challenged* decisions.
11. *Depends on*-relationships cannot originate from *idea* decisions.
12. *Excluded by*-relationships cannot point to *idea*, *tentative*, *discarded* or *rejected* decisions.

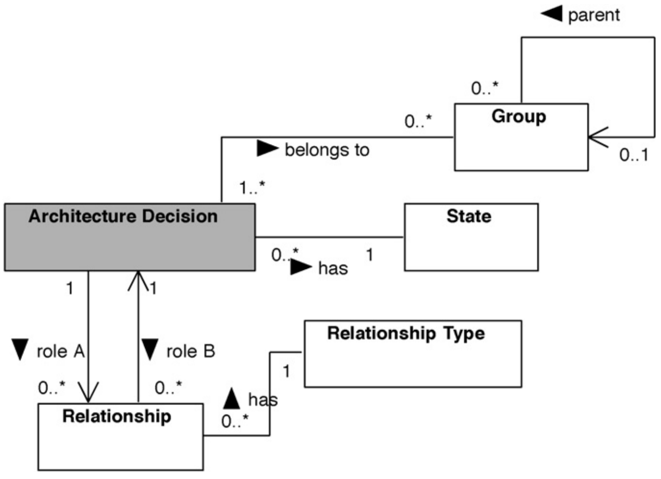


Fig. C.2. Metamodel of Decision Relationship viewpoint.

Table C.1
Mapping of relationship types to Kruchten's ontology.

Used type	Kruchten's type
Caused by	No match
Depends on	Enables ←
	Decomposes
	Subsumes ←
	Comprises ←
Replaces	No match
Is alternative for	Is an alternative to
Is excluded by	Forbids ←
No match	Conflicts with
	Constrains ←
	Overrides

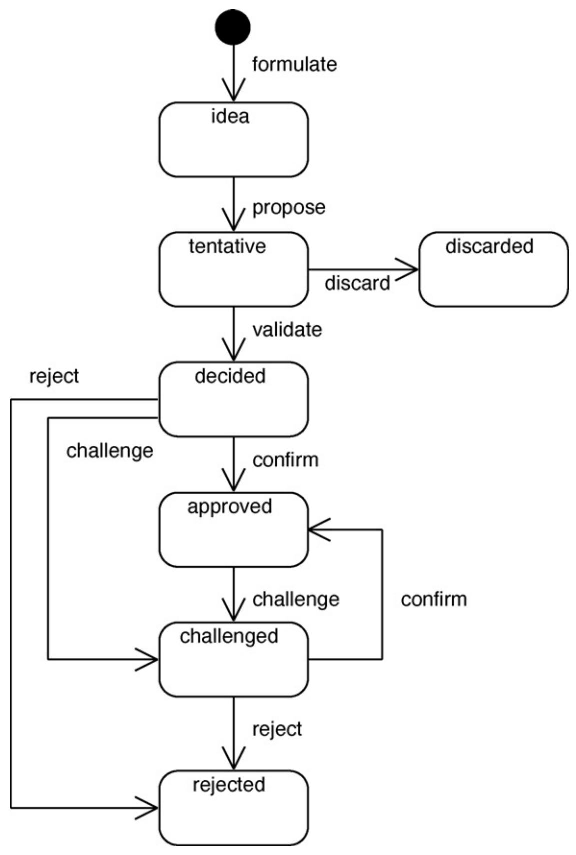


Fig. C.3. UML diagram for state changes of architecture decisions.

13. *Excluded by*-relationships can only originate from *idea*, *tentative*, *discarded* or *rejected* decisions.
14. *Replaces*-relationships can only point to *rejected* decisions.
15. *Replaces*-relationships cannot originate from *idea* decisions.
16. *Alternative for*-relationships cannot point to *idea* or *discarded* decisions.
17. *Alternative for*-relationships can only originate from *tentative* or *discarded* decisions.

Note that the presented constraints refer to the used decision states and relationship types. If different states or relationship types are used, then the constraints must be revised accordingly. In addition to the internal model constraints presented above, cross-viewpoint correspondence rules exist. These rules will be presented in Appendix C.6.

C.3. Decision Chronological viewpoint

This viewpoint shows the evolution of architecture decisions in chronological order. Table 6 shows the concerns framed by this viewpoint related to the respective stakeholders.

C.3.1. Model kind

Fig. C.4 shows a metamodel for the chronological viewpoint. It documents the model kind, which presents the conceptual elements for architecture models that adhere to it. Again, the notation for UML class diagrams is used. An architecture decision is made or changed (i.e., a state change) within an architecture iteration. We define iterations as versions of the architecture as a whole. An iteration endpoint has a date and furthermore a type that can be freely chosen. We propose the following three predefined types:

Milestone: A version of the architecture that has reached a stable state (or an intermediate stable state).
Release: A version of the architecture that is delivered to a customer or made available to the public for use.
Snapshot: A snapshot can be incomplete and possibly inconsistent. This iteration endpoint can be used to express that a customer or project team meeting took place where some decisions were made or discussed without ending up with a stable iteration version.

The following constraints apply to the elements within this model kind:

1. Every decision has a unique name and exactly one state at a time.
2. Every decision can take role A in zero or more relationships (role B is followed by role A).
3. Every decision can take role B in zero or more relationships (role B is followed by role A).
4. Every relationship has the type *followed by*.
5. Every relationship has exactly two non-identical endpoints.
6. Decision states can only change in conformance to the state diagram shown in Fig. C.3.
7. Every iteration has exactly one endpoint with a unique name (e.g. Iteration 4).

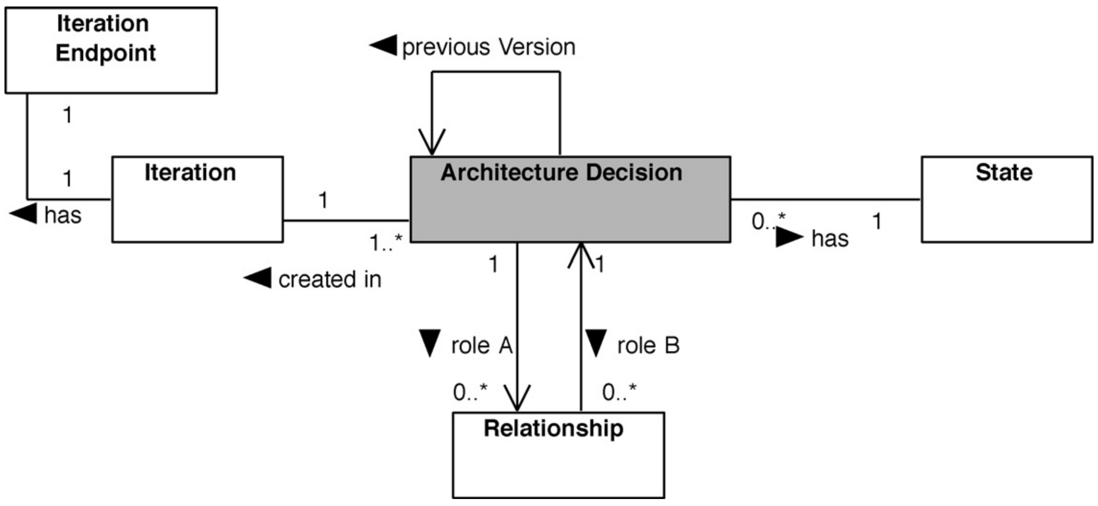


Fig. C.4. Metamodel of chronological viewpoint.

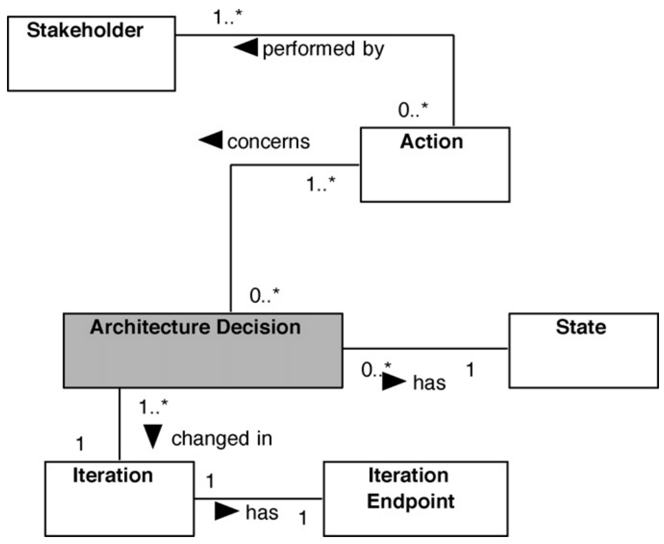


Fig. C.5. Metamodel of stakeholder involvement viewpoint.

8. Concurrent decision paths (e.g. by different architects making decisions autonomously in the same project) cannot cross the boundaries of iterations (marked by an iteration endpoint).

In addition to the internal model constraints presented above, cross-viewpoint correspondence rules exist. These rules will be presented in C.6. An extract of a model that corresponds to this model kind is shown in Fig. 5.

C.4. Decision Stakeholder Involvement viewpoint

The stakeholder involvement viewpoint shows the responsibilities of relevant stakeholders in the decision-making process. Table 5 shows the concerns framed by this viewpoint related to the respective stakeholders.

C.4.1. Model kind

Fig. C.5 shows a meta model for the stakeholder involvement viewpoint. It documents the model kind, which presents the conceptual elements for architecture models that adhere to it. Corresponding elements of the chronological metamodel have the same semantics as in that viewpoint. Every architecture decision is caused by at least one action performed by a stakeholder. In larger projects, the stakeholder can comprise a group or organization, e.g. a development team or a department in a company. The actions can be adapted to the needs of a concrete software project. In our examples we use the following actions:

formulate: A decision is documented as a rough idea that should be revisited in the future. The corresponding decision state is *idea*.
propose: A new decision or a set of new decisions is proposed by an architect. The corresponding decision state is *tentative*.
discard: A tentative decision is discarded by an architect. The corresponding decision state is *discarded*. A discarded decision has never reached a state higher than tentative.
validate: A decision or a set of decisions is validated by a stakeholder. The corresponding decision state is *decided*.
confirm: A decision or a set of decisions is confirmed by a stakeholder on the customer site. The corresponding decision state is *approved*. This action can be performed on a challenged decision to (re-)confirm it or on a decided decision.
challenge: A decision or a set of decisions is challenged by a stakeholder. The corresponding decision state is *challenged*.
reject: A decision that was approved before is rejected. The corresponding decision state is *rejected*.

Fig. C.3 shows stakeholder actions and corresponding decision state transitions.

A stakeholder can have one or more roles in a project. The roles depend on the circumstances in a concrete project. They must be clearly defined prior to being used. We used the following stakeholder roles:

architect: A person or organizational unit responsible for making architecturally relevant decisions in a project.
manager: A person or organizational unit who is responsible for the project in a company.
customer: A person or organizational unit serving as customer representative who is in charge of confirming architecture decisions.

The following constraints apply to the elements of this model kind:

1. Every decision has a unique name.
2. Every iteration endpoint has a unique name.
3. All decision versions changed in one iteration are shown.
4. Only one version of a decision is shown in one model.
5. Every stakeholder has a unique name and zero or more stakeholder roles.
6. Every stakeholder shown performed at least one action.
7. Every action has exactly two non-identical endpoints.
8. Every action originates from exactly one stakeholder in a role.
9. Every action points to a decision, or an iteration endpoint. If the target is an iteration endpoint, then the corresponding action is performed for all decisions (respectively decision versions) changed in that iteration.

In addition to the internal model constraints presented above, cross-viewpoint correspondence rules exist. These rules will be presented in C.6. An example of a model that corresponds to this model kind is shown in Fig. 4.

C.5. Decision Detail viewpoint

The Decision Detail viewpoint provides an in-depth textual description of each architecture decision documented in a software project. Table 3 shows the concerns framed by this viewpoint related to the respective stakeholders.

C.5.1. Model kind

The metamodel for the Decision Detail viewpoint is identical to the shared metamodel for all viewpoints shown in Fig. C.1. In addition to the elements that were already described in the other viewpoint definitions, the model contains a relationship between architecture decision and system concerns. Every architecture decision is represented by exactly one decision detail model. The total of decision detail models shows every architecture decision documented for a system. An example of a model that corresponds to this model kind is shown in Fig. 2.

C.6. Correspondences between viewpoints

The documentation framework for architecture decisions is comprised of four viewpoints. A view conforming to one of these viewpoints is composed of one or more models. The fact that the same subject is represented in multiple independent models creates the risk of inconsistencies. The new ISO/IEC/IEEE 42010 standard for architecture documentation introduces *correspondences* to express cross-model relationships between architecture description elements (ISO, 2011). In the following, we define a number of correspondence rules, which have to be observed by views of the respective viewpoints in order to be consistent. In combination with the correspondence rules, we use a shared metamodel for all model kinds to ensure cross-model consistency. The shared metamodel was introduced in Appendix C.1. The correspondence rules are expressed in terms of constraints and relationships of the architecture models and description elements defined in the metamodel. Note that some of the rules are only applicable if the framework is used as a whole. If the framework or individual viewpoints are customized, then the rules must be revised accordingly.

With the exception of the chronological view, all views are comprised of one or more models. A chronological view comprises one model showing the evolution of all decisions made in the system to document. Please refer to the respective viewpoint sections for more information about internal viewpoint constraints.

R1: The total number of relationship models contains all latest versions of every architecture decisions made in a system. The latest versions must correspond to the latest occurrence of a decision in the chronological model.

R2: A stakeholder involvement model must exist for every iteration shown in the chronological model. Every stakeholder involvement model must contain the versions of architecture decisions belonging to the respective iteration.

R3: A decision detail model contains all incoming and outgoing relationships of a decision shown in the relationship models.

R4: The current state of a decision in the decision detail model must correspond to the state of the latest occurrence of the decision in the chronological model.

R5: The alternatives mentioned in one decision detail model must be identical to the decisions in the relationship view having an *is alternative for* relationship pointing to the decision represented in the model.

R6: The history of a decision represented in the decision detail model must contain all stakeholder actions performed on that decision shown in all stakeholder involvement models.

Appendix D. Example of qualitative analysis process

In Section 4, we described the procedure used to qualitatively analyze parts of the data gathered in the case study. In the following, an example of the analysis process is given. It is taken from the transcript of the focus group conducted after the architecture review. It was chosen because it reflects the typical procedure we used for the qualitative analysis.

Original comment given by one of the domain experts: “I liked the relationship view. I could make use of it quite well. Especially the relationships and what would happen if I changed something. I think this is more clearly illustrated than in any table. This is great progress and I was clearly impressed.”

This passage was labelled with *Research question two* and *relationship view*. It is noticeable that the comment is hard to interpret when taken out of the context. The commenter is referring to the relationship view of the sandnet project, which he was showing to the other participants while talking. He mentions the different relationships between decisions and emphasizes that the relationships can be used to analyze which decisions would be impacted if a specific decision changed. Then he compares the relationship view with a “table”. Here he refers to a decision table, which strictly speaking is a model in a decision detail view. From this comment we derived the following statements:

- Relationship views illustrate the relationships between decisions.
- Relationship views support impact analysis.
- Relationship views illustrate decision relationships better than decision detail views.

Appendix E. Question guide used during the focus group

The following set of questions was used as orientation by the moderator of the focus group, which took place after the review. Please note that the questions were not necessarily asked by the moderator, nor were they answered in a specific order. During an open discussion between the participants, the moderator made sure that the participants gave enough information so that the questions could be answered. The focus group data collection method was described in Section 4.2.2.

- How did the views support the participants in understanding the architecture?
  - Which information were they missing?
  - Which information did they get?
- How did the views help them to communicate architecture, what was missing?
- How do they usually document architecture?
  - What are the liabilities and benefits of the decision views compared to their usual way of doing it?
- Which concerns do they have in architecture documentation in general?
  - How did the relationship view support them, what was missing?
  - How did the chronological view support them, what was missing?
  - How did the documented decisions support them, what was missing?
- Which concerns do the participants have in architecture documentation when starting a new project?
  - How did the relationship view support them, what was missing?
  - How did the chronological view support them, what was missing?
  - How did the documented decisions support them, what was missing?
- Which concerns do they have in architecture documentation when doing architecture reviews? For identifying decisions/sensitivity points/trade-off points and risks?
  - How did the relationship view support them, what was missing?
  - How did the chronological view support them, what was missing?
  - How did the documented decisions support them, what was missing?
- Which concerns do they have in architecture documentation during architecture evolution?
  - How did the relationship view support them, what was missing?
  - How did the chronological view support them, what was missing?
  - How did the documented decisions support them, what was missing?

References

Babar, M.A., Dingsyr, T., Lago, P., van Vliet, H., 2009. Software Architecture Knowledge Management: Theory and Practice. Springer Publishing Company, Incorporated.

Bosch, J., 2004. Software architecture: the next step. In: Software architecture, First European Workshop (EWSA), (3047 of LNCS:194–199).

Capilla, R., Nava, F., Dueñas, J.C.,2007. Modeling and documenting the evolution of architectural design decisions. In: Proceedings of the Second Workshop on SHAring and Reusing architectural Knowledge Architecture, Rationale, and Design Intent. IEEE Computer Society, p. 9.

Clements, P., Garlan, D., Bass, L., Stafford, J., Nord, R., Ivers, J., Little, R., 2002. Documenting Software Architectures: Views and Beyond. Pearson Education.

University of Groningen, Software Engineering and Architecture Group, The Open Pattern Repository. http://code.google.com/p/openpatternrepository/, February 2011.

Dueñas, J., Capilla, R., 2005. The decision view of software architecture. In: Morrison, R., Oquendo, F. (Eds.), Software Architecture, volume 3527 of Lecture Notes in Computer Science. Springer, Berlin/Heidelberg, pp. 88–126.

Glaser, B.G., Strauss, A.L., 1967. The Discovery of Grounded Theory: Strategies for Qualitative Research. Aldine Publishing, New York.

Gray, D.E., 2009. Doing Research in the Real World. Sage Publications Ltd.

IEEE. IEEE Std 1471-2000, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems, October 2000.

IEEE, 2008. IEEE Std 1028-2008, IEEE Standard for Software Reviews and Audits.

ISO, May 2011. Systems and Software Engineering – Architecture Description. ISO/IEC/IEEE 42010, pp. 1–46.

Jansen, A., Bosch, J.,2005. Software architecture as a set of architectural design decisions. In: Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture. IEEE Computer Society, pp. 109–120.

Jansen, A., van der Ven, J., Avgeriou, P., Hammer, D.K.,2007. Tool support for architectural decisions. In: Proceedings of the Sixth Working IEEE/IFIP Conference on Software Architecture. IEEE Computer Society, p. 4.

Kontio, J., Lehtola, L., Bragge, J.,2004. Using the focus group method in software engineering: obtaining practitioner and user experiences. In: Proceedings of the 2004 International Symposium on Empirical Software Engineering. IEEE Computer Society, pp. 271–280.

Kruchten, P., Lago, P., van Vliet, H., 2006. Building up and reasoning about architectural knowledge. In: Hofmeister, C., Crnkovic, I., Reussner, R. (Eds.), Quality of Software Architectures, volume 4214 of Lecture Notes in Computer Science. Springer, Berlin/Heidelberg, pp. 43–58.

Kruchten, P., Capilla, R., Dueñas, J.C., 2009. The decision view's role in software architecture practice. IEEE Software 26 (2), 36–42.

Kruchten, P., 1995. The 4+1 View Model of architecture. IEEE Software 12 (6), 42–50.

Kruchten, P., 2004. An ontology of architectural design decisions in software intensive systems. In: Proceedings of the 2nd Groningen Workshop on Software Variability, pp. 54–61.

Lee, L., Kruchten, P.,2008. A tool to visualize architectural design decisions. In: Proceedings of the 4th International Conference on Quality of Software-Architectures: Models and Architectures. Springer-Verlag, pp. 43–54.

Lethbridge, T.C., Sim, S.E., Singer, J., 2005. Studying software engineers: data collection techniques for software field studies. Empirical Software Engineering 10, 311–341.

Liang, P., Avgeriou, P.,2009. Tools and technologies for architecture knowledge management. In: Software Architecture Knowledge Management: Theory and Practice. Springer, pp. 91–111.

Liang, P., Jansen, A., Avgeriou, P., February 2009. Knowledge Architect: A Tool Suite for Managing Software Architecture Knowledge. Technical Report RUG-SEARCH-09-L01, SEARCH Group, University of Groningen, The Netherlands.

Mack, N., Woodsong, C., MacQueen, K.M., Guest, G., Namey, E., 2005. Qualitative Research Methods: A Data Collector's Field Guide. FLI.

MacLean, A., Young, R.M., Bellotti, V.M.E., Moran, T.P., 1991. Questions, options, and criteria: elements of design space analysis. Human–Computer Interaction 6 (3), 201–250.

University of Groningen, Software Engineering and Architecture Group, The Open Decision Repository. http://opendecisionrepository.googlecode.com, February 2011.

Perry, D.E., Wolf, A.L., 1992. Foundations for the study of software architecture. ACM SIGSOFT Software Engineering Notes 17 (4), 40–52.

Robson, C., 2002. Real World Research: A Resource for Social Scientists and Practitioner-Researchers. Wiley–Blackwell.

Rossow, C., Dietrich, C.J., Bos, H., Cavallaro, L., van Steen, M., Freiling, F.C., Pohlmann, N., 2011. Sandnet: network traffic analysis of malicious software. In: Building Analysis Datasets and Gathering Experience Returns for Security.

Rozanski, N., Woods, E., 2005. Software Systems Architecture: Working with Stakeholders using Viewpoints and Perspectives. Addison-Wesley Professional.

Runeson, P., Hoest, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 14, 131–164.

Seaman, C.B., 1999. Qualitative methods in empirical studies of software engineering. IEEE Transactions on Software Engineering 25 (4), 557–572.

Shahin, M., Liang, P., Khayyambashi, M.R.,2009. Architectural design decision: existing models and tools. In: European Conference on Software Architecture. WICSA/ECSA 2009. Joint Working IEEE/IFIP Conference on Software Architecture, 2009. IEEE, pp. 293–296.

Shahin, M., Liang, P., Khayyambashi, M.R.,2010. Improving understandability of architecture design through visualization of architectural design decision. In: Proceedings of the 2010 ICSE Workshop on Sharing and Reusing Architectural Knowledge. ACM, pp. 88–95.

Stake, R.E., 1995. The Art of Case Study Research. Sage Publications, Inc.

Tang, A., Avgeriou, P., Jansen, A., Capilla, R., Ali Babar, M., 2010. A comparative study of architecture knowledge management tools. Journal of Systems and Software 83 (3), 352–370.

Tyree, J., Akerman, A., 2005. Architecture decisions: demystifying architecture. IEEE Software 22 (2), 19–27.

Williams, R.C., Pandelios, G.J., Behrens, S.G., 1999. Software Risk Evaluation (SRE) Method Description (Version 2.0). Technical Report CMU/SEI-99-TR-029, ESC-TR-99-029, Software Engineering Institute, Carnegie Mellon University, December 1999.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A., 2000. Experimentation in Software Engineering: An Introduction. Kluwer Academic Publishers.

Wohlin, C., Hoest, M., Henningsson, K., 2003. Empirical research methods in software engineering. In: Conradi, R., Wang, A. (Eds.), Empirical Methods and Studies in Software Engineering, volume 2765 of Lecture Notes in Computer Science. Springer, Berlin/Heidelberg, pp. 145–165.

Yin, R.K., February 2003. Case Study Research: Design and Methods, Applied Social Research Methods Series, vol. 5, third edition. Sage Publications, Inc.

Uwe van Heesch is a lecturer for Software Engineering at the Fontys University of Applied Sciences in Venlo, the Netherlands. He is currently pursuing a Ph.D. at the Software Engineering research group at the University of Groningen, the Netherlands. His primary research domain is software architecture, particularly architecture decision management and architecture design reasoning. He is also an active member of the European software patterns community.

Dr. Paris Avgeriou is Professor of Software Engineering in the Department of Mathematics and Computing Science, University of Groningen, the Netherlands where he has led the Software Engineering research group since September 2006. Before joining Groningen, he was a post-doctoral Fellow of the European Research Consortium for Informatics and Mathematics (ERCIM). He has participated in a number of national and European research projects directly related to the European industry of Software-intensive systems. He has co-organized several international workshops, mainly at the International Conference on Software Engineering (ICSE). He sits on the editorial board of Springer Transactions on Pattern Languages of Programming. He has published more than 90 peer-reviewed articles in international journals, conference proceedings and books. His research interests lie in the area of software architecture, with strong emphasis on architecture modeling, knowledge, evolution and patterns.

Rich Hilliard is a freelance software architect and software engineer. He's also editor of ISO/IEC 42010, Systems and Software Engineering—Architecture Description (the internationalization of the widely used IEEE Std 1471:2000). Hilliard is a member of the IFIP Working Group 2.10 on software architecture, the IEEE Computer Society, and the Free Software Foundation, and is an officer of the League for Programming Freedom.